COMPARISON OF EXECUTION TIMES OF AKS ALGORITHM AND MILLER-RABIN ALGORITHM

Research Report

Sai Prashanth Josyula
9207287757
sajo14@student.bth.se

Jaya Krishna Raavi
9307055492
jara14@student.bth.se

Ram Kumar Chilla
930726P652
rach14@student.bth.se

I. Group Members' Participation

The group members participated in idea creation and in report writing with the following amount of involvement.

Group Member	Idea Creation	Report Writing
Sai Prashanth	50 %	50%
Jaya Krishna	45 %	45%
Ram Kumar	5%	5%

***Abstract*—RSA algorithm (Rivest-Shamir-Adleman algorithm) [1] is the most widely used public-key cryptographic algorithm. The RSA key generation phase requires the use of a primality testing algorithm. AKS algorithm (Agrawal–Kayal–Saxena algorithm) is the only deterministic primality testing algorithm that has been proved to have a polynomial time bound [2]. In this paper, we compare the execution times of the AKS algorithm with the state-of-the-art primality testing algorithm, the Miller-Rabin algorithm [3]. We conducted a controlled experiment to record and compare the execution times of the AKS and the Miller-Rabin algorithms. The results of our comparison indicate that the Miller-Rabin algorithm executes significantly faster than the AKS algorithm. In a practical scenario where faster RSA encryptions are required, the use of AKS algorithm is not feasible as it takes a lot of time.**

II. Introduction

Public-key cryptographic algorithms are based on difficult mathematical problems that have inefficient solutions. The RSA algorithm is the most widely used public-key cryptographic algorithm. The RSA algorithm consists of three main phases: key generation, encryption and decryption. The initial step of the key generation phase requires large random prime numbers. The security of the data encrypted with the RSA algorithm rests in part on the difficulty of factoring large numbers [1].

Primality testing algorithms determine whether a given number is a prime number. These algorithms are either probabilistic or deterministic. M. Agrawal, N. Kayal, and N. Saxena present an unconditional deterministic polynomial-time algorithm that determines whether an input number is a prime number [2]. The algorithm is known as AKS algorithm. Miller-Rabin algorithm (M-R algorithm) is the mainstream primality testing algorithm that is based on probability theory [3]. It is the widely used primality testing algorithm in the key generation phase of RSA. It is also the recommended algorithm in Digital Signature Standard (DSS) by The National Institute of Standards and Technology (NIST) [4].

The efficient generation of public-key parameters is a prerequisite in public-key cryptosystems [5]. In the RSA cryptosystem, the RSA modulus is a common parameter for both the public and the private keys. The generation of RSA modulus requires large random prime numbers. A primality testing algorithm is used to generate such numbers. M-R algorithm is the state-of-the-art primality testing algorithm. The use of M-R algorithm in RSA key-generation phase renders the encryption susceptible to attacks [6][7]. Also, the M-R algorithm produces only a probabilistic result. In order to obtain a deterministic result and to make the RSA algorithm more secure against attacks, another appropriate primality testing algorithm should be chosen. The choice of the primality testing algorithm is crucial as it influences the run time of the RSA key generation phase and affects the strength of the RSA encryption.

AKS algorithm can be a possible alternative to the M-R algorithm. The other alternatives are other primality proving algorithms. We chose the AKS algorithm to be the alternative as it is the only primality testing algorithm that has proven to produce a deterministic result in polynomial time [2]. We did not choose a probabilistic algorithm as we want to determine the result with certainty. Among the deterministic primality proving algorithms, AKS algorithm is the only one that is proven to have a polynomial time bound [2]. From the characteristics and the time complexity analysis of the AKS algorithm presented in [2], it has the aspects to replace the use of any other probabilistic primality testing algorithm. Hence, we did not choose any other algorithm as a possible alternative.

Few research papers merely quote that AKS algorithm is relatively slow when compared to the M-R algorithm [8] [9] [10]. These papers come to conclusions based on such opinions. The results of such papers can be validated by confirming the claims/opinions.

In this paper, we compare the execution times of the AKS algorithm with the state-of-the-art primality testing algorithm, the M-R algorithm. If the AKS algorithm, which has theoretically better characteristics than the M-R algorithm has comparatively lower execution times in practice, then it may be feasible for the AKS algorithm to replace M-R in RSA key generation phase. This makes the RSA encryption secure from attacks which arise from the use of M-R algorithm. But if the execution times of AKS algorithm are comparatively very high, then it may not be feasible to use it in the RSA key generation phase in practice. This is because practitioners of RSA cryptography require faster RSA key generations. In this case, further research can be carried out on the efficient implementations of the AKS algorithm. The research also provides a chance to overcome the other drawbacks of the use of M-R algorithm in RSA key generation.

Our research is based on quantitative research methodology. The quantitative research methodology is used for collection of quantitative data from controlled experiments and is well suitable for comparison and statistical analysis of the collected data [11]. The actual input is taken as the independent variable. The execution time is the dependent variable. We chose a random sample of ten inputs for each input size in the range of 9 bits to 12 bits. We carried out the experiment with these samples and recorded the running times for both the algorithms. The analysis of the experiment was divided among the group members. The analysis was performed with Microsoft Excel^{TM}. All the treatments and their outcomes for the experiment were recorded in two separate columns in an Excel^{TM} sheet. The collected data was graphically visualized by creating scatterplots in Excel^{TM}. Then, t-test was performed in the same software. The null hypothesis is rejected based on the results of the t-test.

The remaining paper is structured as follows. Section III describes the background and motivation for our work. This includes state of knowledge, the problem and its importance. Section IV describes in detail, the research definition and plan. Section V describes the research operation and validity of the collected data. Section VI presents the analysis and interpretation of the collected data. Section VII discusses the results. Finally, a summary of the paper is presented and the paper is concluded in Section VIII. Appendices include tables and figures that represent our results pictographically.

III. Background and Motivation

We conducted a systematic literature review to gather background knowledge on the research area. Through the systematic literature review, we gathered the drawbacks of usage of M-R algorithm in RSA key generation, in order to understand the importance of the problem. These drawbacks are summarized in Table I of Appendix A. We also inspected the pros and cons of the AKS algorithm. By understanding the AKS algorithm, we can choose the best implementation of the algorithm. Through our systematic literature review, we also came to know that several works were done to improve the AKS algorithm [12] and implementations of many improved versions of the algorithm exist. One of the motivations of our research is to overcome the drawbacks of the M-R algorithm in RSA key generation and to thus strengthen the RSA encryption.

Though few research papers merely quote that the AKS algorithm is a slower algorithm [8] [9] [10], no systematic comparison of the AKS algorithm and the M-R algorithm was made in those papers. These research papers which quote that AKS algorithm is relatively slower, come to conclusions based on such opinions. Also, most of these research papers were published many years ago. New implementations of AKS which contradict the claims made in those papers may invalidate the results of the papers. One of the motivations of our research is to verify the validity of the claims made in these papers. In this process, we also verify if the current implementations of AKS are also as slow as claimed many years ago. Through the systematic literature review, we figured out the apt gap in the current research area.

Problem:
The M-R algorithm is the state-of-the-art probabilistic primality proving algorithm and is the most widely used one in RSA key generation. Many types of attacks are possible on the prime generation step of RSA key generation when M-R algorithm is used [7]. In order to further secure the data encrypted with the RSA algorithm from possible attacks, an alternative is to use a different primality proving algorithm.

Importance of the problem:
RSA algorithm is widely used for secure data transmission. The security of the transmitted data depends on the strength of the RSA encryption. A strong RSA encryption is preceded by a robust key generation step. The most widely used algorithm in RSA key generation step is the M-R algorithm. Though it is a fast algorithm, the M-R algorithm has many drawbacks as summarized in Table I of Appendix A. If the AKS algorithm can replace the M-R algorithm, then the RSA data encryption can be made more secure than it is in its current state. But, in order to replace the M-R algorithm, the AKS algorithm should have comparatively faster/equal execution times. Also depending on the comparison, further research can be carried out on improvement of implementation of the required algorithm.

IV. Research Definition and Plan

Research Objectives:

- To compare the execution times of the AKS algorithm with the state-of-the-art algorithm, the M-R algorithm.
- To find the effect of the use of AKS algorithm in RSA key generation phase on the speed of the phase.

Research Questions:
The goal of the research is to find answers to the following research questions:

RQ1: How do the execution times of the M-R and the AKS algorithms, which are a measure of the performance of the algorithm, relate to each other?

RQ2: Based on the comparison of the execution times of the algorithms, how does the use of AKS algorithm as an alternative to M-R algorithm for primality testing, affect the speed of the RSA key generation phase?

We perform an experiment to find answers to the research questions. The goal of our experiment is to compare the execution times of both the primality testing algorithms by investigating the following null hypothesis:

H_0: The average time for primality testing is equal for both the algorithms, M-R and AKS.

The alternative hypothesis is formulated below:

H_A: The average time for primality testing is significantly different for both the algorithms, M-R and AKS.

Before designing our experiment, we first select the variables. We select dependent variable/s before selecting the independent variable/s. Based on the domain knowledge, we then choose the appropriate independent variable/s.

Terminology:

Input Size: The algorithms M-R and AKS both being primality testing algorithms, test whether an input value is a prime. In our study, the input size is the number of bits in the binary representation of the input value. For our study, the input size varies in the set {9 bits, 10 bits, 11 bits, 12 bits}. For convenience and ease of understanding, we analyzed the data by taking the decimal value of the input values.

Execution Time: Execution time is defined as the time taken for computing the task. The total time taken to run the program is not considered as the execution time. For example, time taken to determine the primality of the given number, not the time to complete the entire primality test application.

Performance: In our experiment design, we define "performance" in terms of execution times. Lesser execution times of an algorithm indicate better performance of the algorithm. Algorithm with lesser execution times performs better than the one with larger execution times.

Dependent variable:

We derive the dependent variable directly from the formulated hypotheses. The dependent variable is the execution time. After the choice of the dependent variable, we determined the measurement scale of the dependent variable.

Independent variables:

In an experiment, we should be able to control and change the independent variables. Also, the independent variables should have some effect on the dependent variable. We choose the input value in decimal notation as the only independent variable. Input value is chosen to be the independent variable as it can be easily controlled and it has an effect on the dependent variable (execution time). After the choice of the independent variable, we determined the measurement scale of the independent variable.

Measurement Scales:

The measurement scale chosen for both the dependent variable as well as the independent variable is the ratio scale. The other measurement scales are nominal scale, ordinal scale and interval scale. Ratio scale is the most precise and powerful of all the above mentioned types of scales. Logical and mathematical operations can be performed meaningfully on the values measured on a ratio scale. We need a precise measurement of our dependent and independent variables. We also intend to perform a parametric t-test after collecting the values of the dependent variables obtained from the experiment. We thus justify the choice of a ratio scale. We directly measure the dependent variable and the obtained value is an objective measure of the variable. The units of the dependent variable are "seconds".

Confounding variables:

A confounding variable is an extraneous variable that correlates with both the dependent variable and the independent variable. These variables are a threat to the validity of causal occurrences. In our experimental design, memory consumption is a confounding variable. The reason for this is because the memory consumption varies with the input value given to the algorithm (independent variable). The memory consumption also has an effect on the execution time (dependent variable). In our experiment, we do not study this confounding variable.

Research method:

Our research method is a controlled experiment. Other possible alternatives for the choice of the research method include a survey, case study etc. The comparison of various factors for a case study and experiment are given in Table II of Appendix B. The comparison of these factors shows us that for our study, an experiment is a better choice than case study. Similarly, other research methods are not suitable to compare the performance of two algorithms. That is the reason for excluding them as our research methods. Our choice of the research method allows us to accurately compare the performance of both the algorithms under consideration. A reproducible experiment is required for validation of claims. Validation is necessary to establish useful and credible results [13].

Data Collection methods:

In this section, we explain the methods used to generate the input data and collect the output data for the experiment. Random input values were generated for a corresponding chosen input size. The process of generating random input values and recording execution times was implemented in the software. Therefore, each run of the implementation of an algorithm produced an output containing:

- The input value (i_n)
- A binary digit showing whether the input value is prime/composite (P)
- The corresponding execution time (t_n)

If the value of P is 1, it means that the input value is a prime. When P equaled 1, the output data was collected as pairs (i_n , t_n), where i_n represents the n^{th} random input value and t_n represents the corresponding execution time in seconds. For each input size, ten such pairs were collected. This process was repeated for input sizes varying in the set {9 bits, 10 bits, 11 bits, 12 bits}. Thus, after the experiment on both the algorithms, we have 80 pairs of (i_n , t_n) values.

Data analysis methods:

The collected numeric data was described graphically using descriptive statistics. The data sets were described as scatterplots. We checked the scatterplots for any possible outliers and errors. We used scatterplots to describe the data sets. Each graph is good for describing specific properties of a sample. The scatter plot is a good visualization technique to assess the dependencies between variables. Other graphs like box plots, histograms, pie charts etc. are used to efficiently describe certain other properties of a sample. Hence, we used scatterplots for description of our data sets. We then tested the hypothesis using the parametric test, t-test. While certain assumptions are true, the use of parametric methods produce very accurate and precise estimates. We observe that our experimental design and the collected data satisfy the assumptions underlying a parametric test. Instead, the use of a non-parametric test requires a larger sample set to draw conclusions with same degree of confidence. t-test is one of the most commonly used parametric tests and also fairly robust to moderate deviations from preconditions [11]. As our experiment design uses randomly generated inputs, we used the t-test instead of a Paired t-test. From the result of the t-test, we intend to reject the null hypothesis.

V. Research Operation

Operation:

As experimenters, we set up an apparatus and used it to collect data. We then analyzed the data to sustain or contradict hypotheses. We carried out our research operation according to the steps mentioned in [11]. The research operation phase of the experiment consists of three steps: preparation, execution and data validation.

A. *Preparation and Execution*

Necessary preparations were made before the experiment was actually executed. The ease of execution of the experiment depends on how well we perform the preparations.

<u>Experimental Environment:</u> We discuss the workloads, software, hardware, and important parameters used in this experiment. The following are the preparation and execution details of our experiment:

Workloads

We performed the experiment with the sizes of independent variables (input sizes) varying from 9 bits to 12 bits. The specific sizes for the independent variables were chosen due to time constraints. For each input size, ten random values of independent variables were given as input values to an algorithm. Overall, 40 different input values were provided to each algorithm.

Software

We chose the implementation of M-R that makes use of the GNU Multiple Precision Arithmetic Library (GMP). The code was taken from the public domain [14]. The implementation is in C language. The GMP library is designed to be as fast as possible, both for small operands and for huge operands [15]. This increase in speed is achieved by using highly optimized assembly code for the most common inner loops for a lot of CPUs. Thus, we justify the use of GMP library.

Our implementation of AKS algorithm is similar to the efficient implementation in [16]. The implementation is in C++. We did not use the GNU Multiple Precision Arithmetic Library (GMP) while implementing the algorithm. The reason for this being, the use of GMP in an implementation of AKS makes it slower. The GMP implementation uses most of the time in initialization of objects and in the function calls *malloc()* and *free()* [16].

We decided not to implement the programs on our own, from the algorithms. Due to the time constraints, implementing our own programs from the algorithms would give us less time to test the code and the resultant implementations may have some minor/major bugs. Instead, choosing already well-tested implementations helps us to avoid some of the validity threats and assures the quality of our implementations. The implementation parameters are summarized in Table III.

Hardware

The chosen implementations of both the algorithms are compiled and executed on our personal computer. The configuration parameters of the personal computer are summarized in Table IV.

TABLE III. Implementation Parameters

Parameter	Specification
Programming languages	C language, C99 standard C++ language, ISO/IEC 14882:2003 standard
Compilers	gcc 4.8.2 g++ 4.8.2
Libraries	GNU Multiple Precision Arithmetic Library (for M-R algorithm)

TABLE IV. Platform Parameters

Parameter	Specification
CPU	Intel Core i5 2430M
Speed	2.40GHz
RAM	6.00GB Dual-Channel DDR3 @ 665MHz (9-9-9-24)
Motherboard	Hewlett-Packard 166E
Chipset	Sandy Bridge 32nm
Operating System	Windows 7 Home Premium 64-bit SP1
GPU Card	Intel HD Graphics Family (HP) 1024MB ATI Radeon HD 7450M (HP)

Other alternatives for the choice of hardware platform include a high performance computer and a parallel cluster of computers. We exclude the choice of a high performance computer as an alternative. This is because the RSA algorithm is widely used on personal computers for data encryption. Every hardware platform where RSA encryption is carried out will not have the power of a high performance computer. In order to be able to generalize the results to a certain extent, we do not consider that alternative.

We also exclude the choice of a parallel cluster of computers as an alternative. Programs intended to run on a parallel cluster of computers require the use of specific libraries (e.g. MPI, OpenMP). Also, as parallel clusters are not present with every end user of RSA algorithm, we exclude this choice of hardware platform for performing our experiment.

Our decision of the choice of the hardware platform allows the practitioners to reproduce the experiment and verify the results.

Important Parameters

The parameter considered as the basis for comparison is the execution time. The other parameters like memory consumption, power consumption for a run of the program, the difficulty of implementing the algorithm, portability of the implementation on various platforms are not considered. These parameters are not as significant as time consumption of the program. Including too many parameters for the basis of comparison may increase the threats to the validity of our experiment.

The hardware configuration of our personal computer is as mentioned in Table IV. The appropriate software was installed on the computer. We then conducted the experiment on our personal computer. The experiment is conducted in the following stages:

1. We install the necessary software on the hardware platform and compile the AKS and M-R implementation files to create executable files.
2. We specify the input size as a parameter to the AKS executable file. The random input value that was generated is printed on the display screen along with the primality of the input value (prime/composite) and the measured execution time.
3. If the input value is a prime, then we collect the data from the experiment to the data collection sheet. This accounts to a single run of the executable. If the input value is not a prime, then we do not count the run of the executable.
4. For a specific input size, the AKS executable is run ten times. The same process is carried out for the other three input sizes. Therefore, for all the four input sizes, the AKS executable is run 40 times.
5. Similarly, we run the M-R executable for various input sizes, for a total of 40 times.
6. We now have all the appropriate values in the data collection sheet.

We then carried out the analysis and interpretation of the collected data as shown in Section VI.

<u>Data Collection:</u> In this section, we explain the method in which the results of the experiment were collected. We collected the data in a Microsoft Excel^{TM} sheet. We used Microsoft Excel^{TM} as it is convenient to both collect the data and perform statistical tests on it. All the values taken by the independent variable were recorded in one column and values of their corresponding dependent variable were recorded in another column. Graphical representation of the collected data and statistical analysis were also performed in Microsoft Excel^{TM}.

B. Quality Assurance:

The quality of the collected data is validated by addressing the validity threats. We state the validity threats observed till now and how they have been mitigated.

<u>Threats to Conclusion Validity:</u>

- *Violated assumptions of statistical tests:* We made sure that the population being studied satisfies the assumptions of the statistical test chosen for testing the hypothesis. We chose the parametric t-test to test the hypothesis. We can thus assure that we have not violated any assumptions of statistical tests.

- *Unreliability of measures:* The values of the dependent variables for various treatments are measured by the Central Processing Unit (CPU). The implementation for both the algorithms implicitly contains the code required to measure the dependent variable. As the measures are not taken by the experimenter, there is little scope for measurement error. The reason is because the implementation code is well-tested and peer reviewed. We can thus assure the reliability of the measurements.

<u>Threats to Internal Validity:</u>

- *History:* Before performing the experiment, we ensured that only the most essential processes were running on our hardware platform. We terminated any other background process. Thus, after a measurement, we diminish the chance of any other event affecting the next measurement.

- *Selection bias:* The values for the independent variable were randomly generated before recording the execution times for both the algorithms. A treatment is one particular value of an independent variable [11]. The implementation for both the algorithms implicitly contains the code required to randomly generate the treatments. Due to random generation of independent variables, the treatments for both the algorithms form equivalent groups and we can assure that there is no bias in selection of treatments.

<u>Threats to Construct Validity:</u>

- *Inadequate preoperational explication of constructs:* We sufficiently define the constructs before performing the experiment. We ensure that the experiment definition is sufficiently clear to reproduce the experiment.

VI. Data Analysis and Interpretation

Reference [11] describes the three steps involved in quantitative interpretation of the collected experimental data. In the first step, the collected data is characterized using descriptive statistics. In the second step, the data set is reduced by excluding abnormal or false points. In the third step, the data is analyzed by hypothesis testing. We followed the above-mentioned steps to analyze and interpret the collected experimental data.

A. Descriptive Statistics

"Descriptive statistics deal with the presentation and numerical processing of a data set" [11]. After the experimental data is collected, descriptive statistics may be used to describe interesting aspects of the data set and to graphically represent them. Descriptive statistics may be used before carrying out the testing of hypothesis. In our quantitative interpretation, we first characterized the data using descriptive statistics.

We described the data sets by graphical visualization techniques. The reason for using these techniques is that graphs are very illustrative and give a good overview of the data set [11]. We used scatterplots to describe the data sets. We plotted the pairwise samples (i_n, t_n) in two dimensions, where i_n is the nth input value represented as a decimal number and t_n is the corresponding execution time taken to run the algorithm on that input. Scatterplots that were plotted for the data collected from each algorithm can be seen in Figure 1 and Figure 2 of Appendix C.

In Figure 3, the horizontal axis represents the values of the inputs in decimal representation. The vertical axis represents the execution time in seconds. The filled dots represent the execution times of AKS algorithm for various inputs and the unfilled dots represent the execution times of the M-R algorithm for the same input set. The pattern of the AKS algorithm in the figure reflects the polynomial time bound of the algorithm. The pattern of the M-R algorithm indicates the fact that it takes almost equal time for every input value in the chosen data set. A better understanding of the execution times of M-R algorithm can be obtained from Figure 1 of Appendix C.

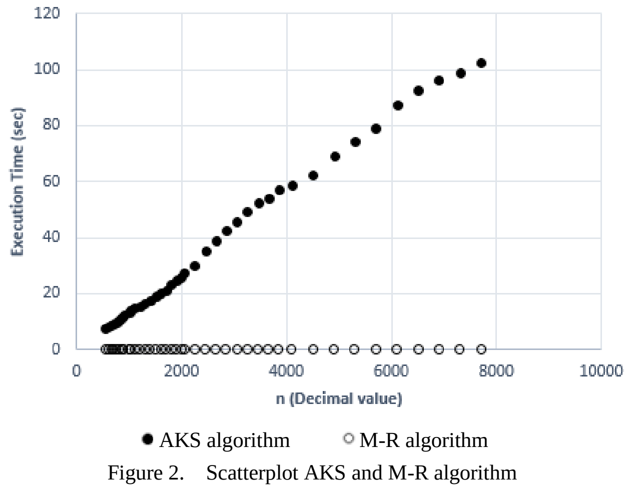


Figure 2. Scatterplot AKS and M-R algorithm

By graphically visualizing the collected data, we now have a good overview of the data set.

B. Data Set Reduction

The results of the statistical methods used in Hypothesis testing depend much on the quality of the input data. Thus, the errors in the collected data sets should be detected and removed. We checked the scatterplots of both the algorithms for possible outliers. We did not exclude any values from the collected data sets.

C. Hypothesis Testing

"The objective of hypothesis testing is to see if it is possible to reject a certain null hypothesis, H_0, based on a sample from some statistical distribution" [11]. Our aim is to reject the null hypothesis. We used a parametric test to reject the null hypothesis. We used the t-test to reject the null hypothesis.

- **t-test**

The t-test is a parametric test that is used to compare two independent data samples. We performed the t-test on our samples with the assumptions as mentioned in [11]. The test is performed as given in Figure 4 of Appendix D.

TABLE V. T-TEST

Input	Two independent samples of execution times, for instance, AKS sample: (7.9, 8.4, 8.9, …, 96.5, 99.1, 102.7) M-R sample: (0.00E+00, 1.00E-07, …,1.25E-07)
H_0	$\mu_x = \mu_y$ i.e. the expected mean values are the same
Calculations	$S_p = \frac{\sqrt{(40-1)\,(869.28) + (40-1)(2.76\text{E}-15)}}{(40+40-2)}$ $t_0 = \frac{(39.0225) - (8.25\text{E}-08)}{S_p\sqrt{\frac{1}{40}+\frac{1}{40}}} = 8.371$
Criterion	Two sided $(H_1: \mu_x \neq \mu_y)$: **Rejected H_0** , Because $\lvert t_0 \rvert > t_{\alpha/2,\, n+m-2}$ 8.371 > 2.023

A lower p-value is a strong evidence that the null hypothesis does not hold true. The p-value obtained from the t-test is 3.05152E-10. The other details of the t-test are mentioned in Table VI and Table VII of Appendix E. The criterion considered for rejecting the null hypothesis is shown in Table V. As the value of t_0 is greater than the value of $t_{\alpha/2,\ n+m-2}$, we rejected H_0.

By summarizing, aggregating and analyzing the collected data, we answer our research questions.

RQ1: How do the execution times of the M-R and the AKS algorithms, which are a measure of the performance of the algorithm, relate to each other?

From the graphical visualization of collected data, we can observe that the execution times of AKS algorithm are very slow compared to those of the M-R algorithm. From the results of the parametric t-test, we rejected the null hypothesis. Thus, our results indicate that for the presented experimental setup, the execution times of the M-R algorithm are significantly less than the execution times of the AKS algorithm. From our definition of performance of an algorithm, we can also say that the M-R algorithm performs very well compared to the AKS algorithm.

RQ2: Based on the comparison of the execution times of the algorithms, how does the use of AKS algorithm as an alternative to M-R algorithm for primality testing, affect the speed of the RSA key generation phase?

The AKS algorithm took larger execution times for even small inputs that were computed upon significantly faster by the M-R algorithm. The main step in the RSA key generation phase is the generation of a large random prime number. The speed of the RSA key generation step is proportional to the speed of the prime generation step. The generation of prime numbers is carried out by testing large random numbers for primality. Hence, the speed of RSA key generation phase is proportional to the speed of primality testing algorithm used in the phase. The results of our experiment indicate that the AKS algorithm is significantly slower than the M-R algorithm. Hence, the use of AKS algorithm for primality testing significantly increases the time required to complete the RSA key generation phase.

VII. Discussion

A. Contributions

The results of our comparison show that for our experimental setup, execution of the AKS algorithm implementation takes more time than the M-R algorithm implementation. There is a significant difference in performance of both the algorithms. The use of AKS algorithm in RSA key generation phase has a potential to improve the strength of RSA encryption by overcoming the drawbacks of M-R algorithm. But the observed significant difference in performance excludes the use of the AKS algorithm as an alternative to M-R algorithm in RSA key generation. Though, in cases where RSA key generation phase is performed infrequently, the AKS algorithm can be used, thus resulting in a strong RSA encryption.

Our study validates that the execution times of AKS algorithm are significantly slower than those of the M-R algorithm. Our study confirms the claims in the papers that merely state their opinion about the AKS algorithm being slow and which do not systematically prove it. Through our experiment, we systematically proved that the AKS algorithm is slower than the M-R algorithm.

AKS can be highly parallelized [12]. In our experiment, we have not used this fact during our implementation. In recent years, parallel computing has become a dominant paradigm. Therefore, there is scope to improve the running time of the AKS algorithm, by utilizing its parallelizability. Our results encourage the researchers to focus on improving the implementations of AKS algorithms on parallel systems, so that it performs at least as well as the state-of-the-art algorithm.

Implications for Practice: If the RSA key generation step has to be performed more frequently, then using the state-of-the-art primality testing algorithm in key generation phase is better. If the key generation step is to be performed infrequently and a deterministic result is required, then from our results we can say that the use of AKS algorithm results in overcoming the drawbacks of the usage of M-R algorithm.

B. Threats to Validity

In this section, we discuss the threats to the validity of the results obtained through the experiment. Reference [11] lists four types of threats to the validity of the results of an experiment. They are threats to conclusion, internal, construct and external validity. Different classification schemes exist for different types of threats to the validity of an experiment.

Conclusion Validity: Any issue that affects the ability to draw correct conclusion about relations between the treatment and the outcome of an experiment is a threat to conclusion validity.

- *Low Statistical Power:* The power of a statistical test is its ability to reveal a true pattern in the data [11]. Low power occurs due to small sample sizes. The sample sizes considered in our study are not too large. This is a threat to the conclusion validity of the results of our study.

Internal Validity: The relationship observed between the treatment and the outcome should be a causal relationship. Influences that can possibly affect the treatment with respect to causality, without the knowledge of the researcher are threats to internal validity.

- *Existence of confounding variables:* Confounding variables are those variables excluded in our study, which have a correlation with the dependent as well as the independent variables. The changes observed in the dependent variable may rather be attributed to the variations in the confounding variable which is related to the independent variable. This is a threat to the validity of causal inferences. Memory consumption is a confounding variable that may affect our inferences.

Construct Validity: Construct validity is the extent to which we measure what we claim to have measured. Some of the construct validity threats relate to the design of the experiment, and others to social factors [11].

- *Mono-operation bias:* Our experiment includes only one independent variable, the input value. The experiment may lack a comprehensive view of the theory.
- *Mono-method bias:* In our experiment, we use a single type of observation. The code to record the measures of execution times is implicit in the implementation of the algorithm. The same experimenter made all the observations from the output displayed on the computer and collected them in Microsoft Excel^{TM} sheet. As this process of data collection is a little strenuous, the experimenter may have made some minor errors in the data collection sheet.

- *Restricted generalizability across constructs:* We defined better performance to mean lesser execution times. But, depending on the implementation, lesser execution times of an algorithm may effect memory consumption and reliability of the result in an unintended way. We draw conclusions only depending on the execution times and ignore these other factors in our study.

External Validity: External validity concerns the extent to which the results of our experiment can be generalized. Threats to external validity are factors that limit the generalizability of our experimental results.

- *Threats to Generalizability:* As we implemented the algorithms on our personal computers, we cannot generalize the conclusions for all hardware platforms. The conclusions of our study cannot be generalized for various other implementations of the algorithms (e.g. serial implementations using other libraries, parallel implementations). Also, the chosen data sets may not replicate the entire population. The small input sizes (9 bits, 10 bits, 11 bits, 12 bits) of input values were chosen due to time constraints.

Limitations:

- The results of our study can be held true only for similar hardware platforms and implementations that are chosen for the study. We cannot generalize the results for every hardware platform and very different implementations of the algorithms.
- Each independent variable has an input size in the set {9 bits, 10 bits, 11 bits, 12 bits}. Though the independent variables (input values) were chosen randomly, the input sizes are not random. Thus, our results cannot be generalized for input sizes with very different magnitudes than the chosen ones.
- We did not present a quantitative measure for the answer to *RQ2* i.e. the effect of the use of AKS algorithm on the speed of RSA key generation phase. A quantitative measurement of the effect would have given a better understanding of the effect on the speed.

VIII. Summary and Conclusions

Thus, through our research, we compared the execution times of AKS algorithm with the state-of-the-art algorithm, the M-R algorithm. The execution times of the AKS algorithm for various inputs are significantly larger in magnitude when compared to the M-R algorithm. The use of AKS algorithm in RSA key generation phase significantly increases the time taken to complete the phase. Thus, in a practical scenario where faster RSA encryptions are required, the use of AKS algorithm is not feasible as it takes a lot of time. Further research can be done on implementing efficient parallel versions of the AKS algorithm and improving the running time of the implementation. Future work can also be done on the comparison of memory consumed by the implementations of both the algorithms.

References

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[2] M. Agrawal, N. Kayal, and N. Saxena, "PRIMES is in P," *Ann. Math.*, pp. 781–793, 2004.

[3] M. O. Rabin, "Probabilistic algorithm for testing primality," *J. Number Theory*, vol. 12, no. 1, pp. 128–138, 1980.

[4] Li Dongjiang, Wang Yandan, and Chen Hong, "The research on key generation in RSA public-key cryptosystem," in *2012 Fourth International Conference on Computational and Information Sciences (ICCIS), 17-19 Aug. 2012*, 2012, pp. 578–80.

[5] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.

[6] T. Finke, M. Gebhardt, and W. Schindler, "A New Side-Channel Attack on RSA Prime Generation," in *Cryptographic Hardware and Embedded Systems-CHES 2009*, Springer, 2009, pp. 141–155.

[7] C. Vuillaume, T. Endo, and P. Wooderson, "RSA Key Generation: New Attacks," in *Constructive Side-Channel Analysis and Secure Design. Third International Workshop, COSADE 2012, 3-4 May 2012*, 2012, pp. 105–19.

[8] Q. Cheng, "Primality proving via one round in ECPP and one iteration in AKS," in *Advances in Cryptology-CRYPTO 2003*, Springer, 2003, pp. 338–348.

[9] F. Morain, "Implementing the asymptotically fast version of the elliptic curve primality proving algorithm," *Math. Comput.*, vol. 76, no. 257, pp. 493–505, 2007.

[10] P. Berrizbeitia and A. Olivieri, "A Generalization of Miller's Primality Theorem," *Proc. Am. Math. Soc.*, vol. 136, no. 9, pp. 3095–3104, Sep. 2008.

[11] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer, 2012.

[12] R. E. Crandall and J. S. Papadopoulos, "On the implementation of AKS-class primality tests," *Adv. Comput. Group Apple Comput. E Univ. Md. Coll. Park*, 2003.

[13] W. F. Tichy, P. Lukowicz, L. Prechelt, and E. A. Heinz, "Experimental evaluation in computer science: A quantitative study," *J. Syst. Softw.*, vol. 28, no. 1, pp. 9–18, 1995.

[14] "Miller-Rabin primality test (C, GMP) - LiteratePrograms." [Online]. Available: http://en.literateprograms.org/Miller-Rabin_primality_test_(C,_GMP). [Accessed: 18-Apr-2014].

[15] "The GNU MP Bignum Library." [Online]. Available: https://gmplib.org/. [Accessed: 27-May-2014].

[16] C. Rotella, "An Efficient Implementation of the AKS Polynomial-Time Primality Proving Algorithm," *Sch. Comput. Sci.-Carnegie Mellon Univ. Pittsburgh-Pa.-USA*, 2005.

[17] C. Clavier, B. Feix, L. Thierry, and P. Paillier, "Generating Provable Primes Efficiently on Embedded Devices," in *Public Key Cryptography - PKC 2012. 15th International Conference on Practice and Theory in Public Key Cryptography, 21-23 May 2012*, 2012, pp. 372–89.

[18] Young-Sik Kim, S. R. Shrestha, and Ji-Woong Jang, "Refined Algorithm for Prime Number Generation in Embedded Security Systems," in *2011 IEEE Asia-Pacific Services Computing Conference (APSCC), 12-15 Dec. 2011*, 2011, pp. 406–9.

[19] Chong Fu and Zhi-Liang Zhu, "An efficient implementation of RSA digital signature algorithm," in *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), 12-14 Oct. 2008*, 2008, p. 4 pp.

[20] Chenghuai Lu and A. L. M. Dos Santos, "A note on efficient implementation of prime generation algorithms in small portable devices," *Comput. Netw.*, vol. 49, no. 4, pp. 476–91, Nov. 2005.

Appendix A

TABLE I. Drawbacks of Miller-Rabin Algorithm

Primary study	Drawbacks/Challenges
Ref [4]	To largely decrease the calling times of Miller-Rabin test, a pre-screening algorithm is used. The implementation of this algorithm is an overhead to the user.
Ref [7]	Many types of attacks are possible on the prime generation step of RSA key generation when Miller-Rabin algorithm is used.
Ref [17]	To ensure compliance with the industry standards, many iterations of Miller-Rabin are to be carried out during a primality test. This increases the running time of key generation.
Ref [18]	Miller-Rabin is a probabilistic primality test. Hence, the result is a mere probabilistic conclusion and not deterministic.
Ref [19]	In worst case scenario, the prime number may contain a small prime factor.
Ref [20]	Probabilistic primality tests are computationally expensive. They are the most time consuming part of prime generation.

Appendix B

TABLE II. Comparison of Factors in Case Study And Experiment

Factor	**Experiment**	**Case study**
Level of Control	High	Low
Difficulty of Control	Low	High
Replication Level	High	Low
Cost of Replication	Low	High
Generalizability	Yes (maybe)	No

APPENDIX C

In Figure 3, the horizontal axis represents the values of the inputs in decimal representation. The vertical axis represents the execution time in seconds. From the values on the vertical axis, it can be observed that the highest recorded execution time is 0.125 microseconds. For some inputs, the algorithm runs so fast that the execution time is almost negligible. Such inputs are represented in the figure by plotting the points on the horizontal axis. The pattern in the figure appears irregular. One of the reasons for this irregularity is the nature of the algorithm. The M-R algorithm is a probabilistic algorithm and hence the execution time does not always increase with increase in the value of inputs or input sizes. Another reason for this pattern is the very small value of scale that is taken on the vertical axis. The scale of the vertical axis is 20 nanoseconds and hence any small change in the execution time reflects as a significant change in the pattern in Figure 4.

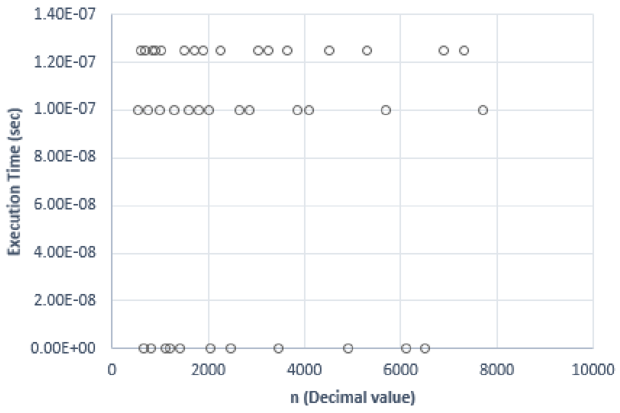


Figure 5. Scatterplot M-R algorithm

In Figure 6, the horizontal axis represents the values of the inputs in decimal representation. The vertical axis represents the execution time in seconds. From the values on the vertical axis, it can be observed that the highest recorded execution time is 103 seconds. The pattern in the figure reflects the polynomial time bound of the algorithm.

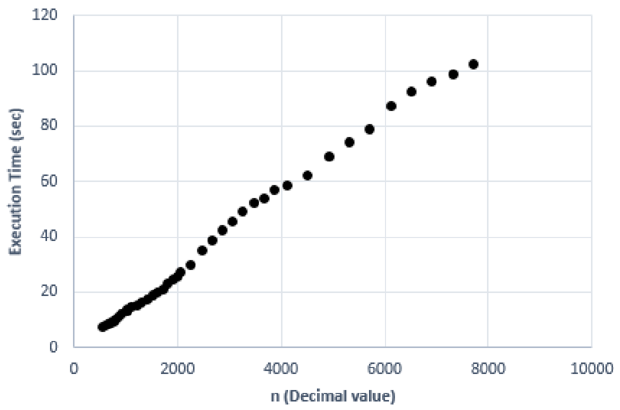
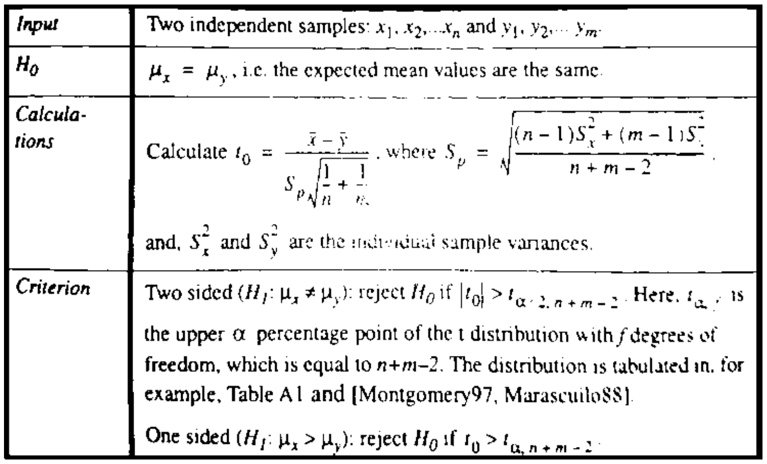


Figure 7. Scatterplot AKS algorithm

APPENDIX D



Input	Two independent samples: $x_1, x_2, \ldots x_n$ and $y_1, y_2, \ldots y_m$.
H_0	$\mu_x = \mu_y$, i.e. the expected mean values are the same.
Calculations	Calculate $t_0 = \frac{\bar{x} - \bar{y}}{S_p \sqrt{\frac{1}{n} + \frac{1}{m}}}$, where $S_p = \sqrt{\frac{(n-1)S_x^2 + (m-1)S_y^2}{n+m-2}}$, and, S_x^2 and S_y^2 are the individual sample variances.
Criterion	Two sided (H_1: $\mu_x \neq \mu_y$): reject H_0 if $\lvert t_0 \rvert > t_{\alpha/2, n+m-2}$. Here, $t_{\alpha, f}$ is the upper α percentage point of the t distribution with f degrees of freedom, which is equal to $n+m-2$. The distribution is tabulated in, for example, Table A1 and [Montgomery97, Marascuilo88]. One sided (H_1: $\mu_x > \mu_y$): reject H_0 if $t_0 > t_{\alpha, n+m-2}$.

Figure 4. Procedure for t-test [11]

APPENDIX E

TABLE VI. t-test: Two-Sample Assuming Unequal Variances (a)

Parameter	Var 1	Var 2
Mean	39.0225	8.25E-08
Variance	869.2792244	2.76282E-15
Observations	40	40

TABLE VII. t-test: Two-Sample Assuming Unequal Variances (b)

Hypothesized Mean Difference	0
df	39
t_0	8.370770477
P(T<=t) one-tail	1.52576E-10
t Critical one-tail	1.684875122
P(T<=t) two-tail	3.05152E-10
t Critical two-tail	2.02269092