e-ISSN: 2721-3013, p-ISSN: 2721-3005
DOI: https://doi.org/10.38035/jafm.v3i6
Received: 14 December 2022, Revised: 26 January 2023, Publish: 19 Feberuary 2023
https://creativecommons.org/licenses/by/4.0/

DINASTI® RESEARCH
JAFM
JOURNAL OF ACCOUNTING AND FINANCE MANAGEMENT
E-ISSN: 2721-3013, P-ISSN: 2721-3005
https://dinastires.org/JAFM
editor@dinastires.org
0811 7401 455

Digital Signature Standard Implementation Strategy by Optimizing Hash Functions Through Performance Optimization

Ferraro S. Octora Ginting[1*], Veithzal Rivai Zainal[2], Aziz Hakim[3]
[1)] Universitas Krisnadwipayana, Jakarta, email: ferraro.gintings@gmail.com
[2)] Chairman of the Board of Trustees of the Indonesian MSME Communication Forum, email: veithzal47@gmail.com
[3)] Universitas Krisnadwipayana, Jakarta, email: dr_azishakim@unkris.ac.id

*Corresponding Author: Ferraro S. Octora Ginting[1]

Abstract: Security method of data transmission process has been growing rapidly with the science of cryptography. Cryptography can provide security services that includes security aspects like confidentiality, data integrity, authentication and non-repudiation. Modern cryptography uses a key that must be kept secret to overcome the problem of cryptographic security. Problem in the use of the same key by two entities that communicate with each other in exchanging messages is a way to distribute the key. This problem can be overcome by using public-key cryptography, which allows users to communicate securely without a shared secret key. Digital signature is the application of public-key cryptography. When accessing important digital documents, it is necessary to verify the signature given. Implementation of digital signature always requires a hash function. Hash function used in this research namely SHA-256, SHA-384 and Tiger. Federal Information Processing Standards (FIPS) set the cryptographic standard for digital signatures is the Digital Signature Standard (DSS). Algorithms included in the DSS are the Digital Standard Algorithm (DSA), Ron Rives, Adi Shamir, and Leonard Adleman (RSA) and Elliptic Curve Digital Signature Algorithm (ECDSA). So it is necessary to test the best digital signature implementation strategy that can be used by optimizing the performance of the hash function. Performance testing of the three algorithms is done by making an application using a computer programming language C++. Implementation program using C++ class library for cryptographic scheme that is Crypto++ Library 5.6.0. Class libraries used in the classes functions for digital signatures. On the application of digital signatures generated, conducted tests is done by combining each hash function algorithm with each of the DSS in order to compare their performance in terms of time and memory usage. Against the test results are then analyzed using statistical tests. The result shows that pair of Tiger hash function and DSA algorithm is the best combination.

Keywords: Cryptography, Digital Signatures, Hash Functions, Crypto + + Library, Anova

INTRODUCTION

Cryptography plays an important role in security services such as confidentiality, data integrity, authentication and denial prevention. Modern cryptography uses keys that must be kept secret to overcome cryptographic security problems. The problem of using the same key by two entities that communicate with each other can be overcome by using public-key cryptography, which allows users to communicate securely without the need to share secret keys.

Digital signature is an application of public-key cryptography. A digital signature has a key which consists of a public key and a private key. Implementation of a digital signature always requires a hash function. The hash function produces an output called Message Digest (MD). The Federal Information Processing Standards (FIPS) defines the cryptographic standard for digital signatures as the Digital Signature Standard (DSS). The algorithms included in the DSS are the Digital Standard Algorithm (DSA), Ron Rives, Adi Shamir, and Leonard Adleman (RSA) and the Elliptic Curve Digital Signature Algorithm (ECDSA).

This research examines the suitability of the DSS fitting with the hash function, which is necessary to find the resulting performance effect. This research is a development of previous research conducted by Tamici (2007) from the Bandung Institute of Technology entitled Performance Analysis of Cryptographic Secure Hash Standard (SHS) and Digital Signature Standard (DSS). The development in this study was carried out on the hash function used, namely the addition of the Tiger hash function and the reduction of the hash function which is included in the SHS category. In the previous research, the implementation phase used Matlab 7.0.3 software, while this study used the C++ programming language by utilizing a cryptographic scheme framework, namely (Crypto++ Library 5.6.0).

This study aims to determine the best hash function implementation strategy by first designing an application that combines DSS with a hash function that functions to receive input in the form of files in text, executable and image formats, then proceed by comparing the performance for each test result of combining DSS with functions hash in terms of time and memory usage, and then analyze the variety of test results using statistical tests.

LITERATURE REVIEW

Modern Cryptography

Cryptography is the science and art of keeping messages secure. Modern cryptographic program codes are made so complex that it is very difficult to crack the ciphertext without knowing the key (Munir 2006).

Cryptanalysis

Cryptanalysis is the study of mathematical techniques that try to break cryptographic techniques, while cryptanalysts are people who work on cryptanalysis (Menezes et al. 1996). The work factor for solving the exhaustive search system increases exponentially with increasing key length (Munir 2006).

Public-Key Cryptography

Public-key cryptography (Asymmetric) is an encryption model with a public key and a private key. The key for encryption is announced to the public and is symbolized by e, while the key for decryption is secret and is symbolized by d (Munir 2006).

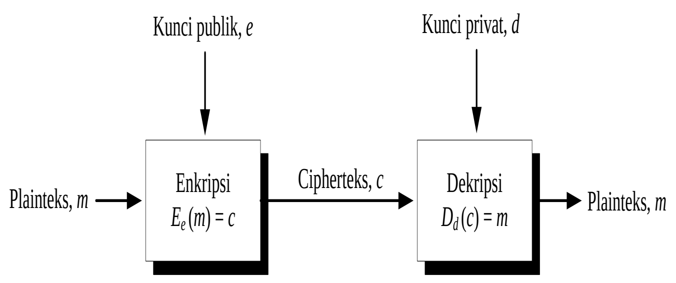

Figure 1. Diagram of Public-Key Cryptography (Munir 2006)

Flow-Lock Generator

The key stream is generated from a generator called a key stream generator (Munir 2006).

Pseudo Random Number Generator

Random numbers are used to generate key elements, generate initialization vectors, generate key parameters in public-key cryptographic systems and so on (Munir 2006).

Hash Function Modeling

A hash function is a function that accepts an input string of arbitrary length and converts it into a fixed length output string (Munir 2006). The output is also called message digest (MD).

One-way hash function (One-way Hash) is a hash function that works in one direction. The general structure of the hash function is shown in Figure 2.

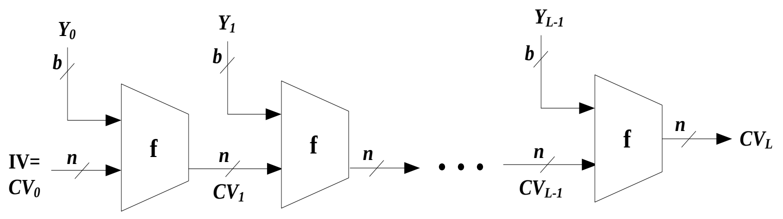

Figure 2. The General Structure of the Hash Function (Stalling 2003)

In this study, only three hash functions were used, namely SHA-256, SHA-384 and Tiger. The specifications of the three hash functions are shown in Table 1.

Table 1. SHA-256, SHA-384 and Tiger Hash Function Specifications

Tipe *hash*	Spesifikasi Ukuran (bits)			
	Pesan	Blok	*Word*	MD
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
TIGER	$< 2^{64}$	512	64	192

Digital Signature

A digital signature is similar in purpose to a handwritten signature. Its purpose is to provide a tool used by entities to tie their identities into a single piece of information (Menezes et al. 1996). The public-key cryptographic system is suitable for digital signatures using a hash function (Munir 2006).

Digital signatures included in the DSS standard, namely:

DSA, cannot be used for encryption, but is specifically for signature formation and checking the validity of signatures. The security level of DSA is based on the computational power of the discrete logarithm in the Zp* prime sequence subgroup. RSA, involves a public key and a private key. The security of the RSA algorithm lies in the difficulty of factoring large numbers into prime factors.

ECDSA uses an elliptic curve analogy, namely the Elliptic Curve Cryptosystem (ECC). This algorithm uses elliptic curve discrete logarithm math problems, the level of security is calculated based on the length of the key.

Public-Key Cryptographic Security Attacks

Munir (2006) argues, based on the technique used, attacks are divided into two, namely:

a. Exhaustive attack or brute force attack
b. Analytical attacks

Analysis of Variety (Anova)

Analysis of variance is a computational test that is applied to data generated by experiments designed on controlled variables. The purpose of the factorial design is to see the interaction between the factors, sometimes they synergize with each other (positive) responses, but sometimes the presence of one factor hinders the performance of other factors (negative) (Setiawan 2009).

In this study, the RAL factorial design was applied. The use of RAL is when the environmental conditions encountered are homogeneous or there are no other factors that influence the response outside of the factors being tried or studied (Walpole 1990).

To facilitate data processing, Minitab statistical software is used. Minitab provides facilities for making statistical graphs easily and displaying them in a more attractive, informative form and at the same time involves the concept of probability (Triyanto 2009).

METHOD

The method used in this research includes literature study, implementation using Crypto++ Library 5.6.0., testing the implementation results using the C++ programming language and preparing research report documentation. The test consists of two components, namely:

a. Testing the results of combining digital signature algorithms with hash functions,
b. Diversity analysis.

The stages of this research are shown in the diagram in Figure 3.

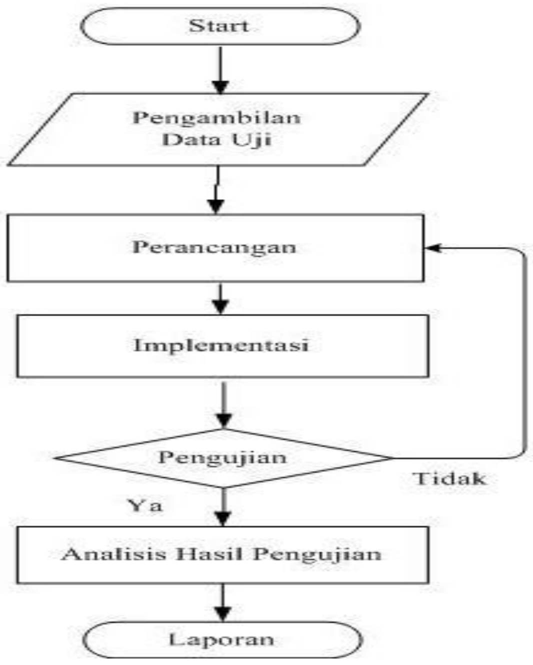

Figure 3. Research stages

RESULTS AND DISCUSSION

Retrieval of Test Data

The data used in this study are files containing characters in text format in the form of letters, numbers and symbols, executable files and input files in the form of images. The file size for a test data is taken as a multiple of the size of the previous test data file. The first test data in this study is 5 MB in size and the maximum test data file size is 80 MB for the fifth test data.

Digital Signature Application Design

The application created has the following main functions:

a. Receive input data files stored on the user's hard disk from a specified directory address.
b. Generate a pair of keys, namely the private key and the public key (key generation).
c. Create a digital signature for a particular file entry (signature generation).
d. Verifying the validity of the owner of the signature or the file being checked (signature verifying).

The implementation of this digital signature application uses the Microsoft Windows operating system. Tests for combining digital signature algorithms with hash functions are made according to the test scenario shown in Table 2.

Table 2. Test Scenarios for Combining Digital Signature Algorithms with Hash Functions

Skenario	Algoritme	Fungsi *Hash*
1	DSA	SHA-256
2	DSA	SHA-384
3	DSA	Tiger
4	RSA	SHA-256
5	RSA	SHA-384
6	RSA	Tiger
7	ECDSA	SHA-256
8	ECDSA	SHA-384
9	ECDSA	Tiger

Implementation of Making Digital Signature Applications

The implementation of digital signature applications is made using the C++ programming language by utilizing an open source library for cryptographic schemes, namely the Crypto++ Library. This application can be run through the command prompt application. The menus used include three process operations namely key pair generation, signature generation and signature verification.

Testing Digital Signature Applications

Testing of digital signature applications is carried out by providing input and observing the output results. In this research, a digital signature application has been tested for 3 processes namely, key generation, signature generation and signature verification with successful test results.

For the key generation process, it produces output in the form of a private key and a public key. The signature generation process produces output in the form of a signature file, processing time and the amount of memory used. The output of the signature verification process is information on the results of verification, i.e. verification has been successfully carried out, as well as processing time and the amount of memory used.

Analysis of Test Results

This section describes the results of testing the digital signature application. The method used to identify general conclusions from the test results for processing time is carried out through statistical tests, while the test results for memory usage are not statistical because the results are quite significantly different. Data analysis of the test results is divided into two parts, namely an analysis of the results of the signing process time and the verification process time, then an analysis of the results of identification of memory usage both during the signing and verifying processes.

In this study, with $\alpha = 5\%$, the 95% confidence level is the minimum number that must be achieved by the data obtained in order to reject H0. The research hypothesis includes H0, namely the treatment means are the same and H1, namely one or more pairs of treatment means are different.

Data processing for analysis of variance was carried out using Minitab software which will be explained further in the section on variance analysis for signing and verifying processing times.

The next analysis is an analysis of the results of identification of memory usage both during the signing and verifying processes. The data that has been recorded is processed using Microsoft Office Excel which is presented in the form of a graph to facilitate observation. This analysis cannot use analysis of variance because the data does not meet the two assumptions of analysis of variance, because the data is not normally distributed and the variance is not homogeneous.

Analysis of the Variation of Signing Process Time and Verifying Process Time

The analysis at this stage is an analysis of the results of tests that compare the performance of digital signature algorithms in general using the SHA-256, SHA-384 and Tiger hash functions at a certain file size which includes the signing process time and the verifying process time. The method used is a factorial design with two factors, namely an algorithm consisting of 3 levels (DSA, ECDSA and RSA) and a hash function factor consisting of 3 levels (SHA-256, SHA-384 and Tiger). Data processing for analysis of variance was carried out using Minitab software.

The two minimum data requirements are stated to meet the assumptions in the analysis of variance are:

a. Data is normally distributed.
b. Variety is homogeneous.

To find out the data from the test results meet these two assumptions, a test is first carried out using the data that has been obtained. In this study, testing was carried out on data during the signing process with a file size of 5Mb. Testing is carried out by carrying out two tests, namely the normal probability plot and the test for equal variances.

Signing Process Time Analysis

Identification of time performance, can also be reviewed from the use of hash functions. The results of the comparison of the signing process time are shown in Figure 5. From these results it appears that the DSA algorithm with the Tiger hash function looks better than the others.

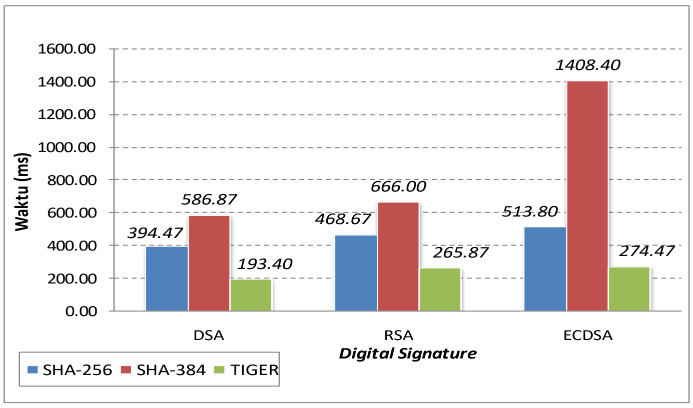

Figure 4. Comparison of the Signing Time for the DSA, RSA and ECDSA Algorithms.

However, these results do not show definitively that the three algorithms are significantly different in terms of performance. A significant difference can also be seen in the use of the hash function. Therefore, an Anova test was carried out.

From the results of the ANOVA test during the signing process, it is known that the resulting P values (probability) are all smaller than α (with α=0.05). These results suggest that the algorithm and hash function make a real difference to the signin process.

Verifying Process Time Analysis

The results of the comparison of the verifying process times are shown in Figure 6. From these results it appears that the DSA algorithm with the Tiger hash function looks better than the others.

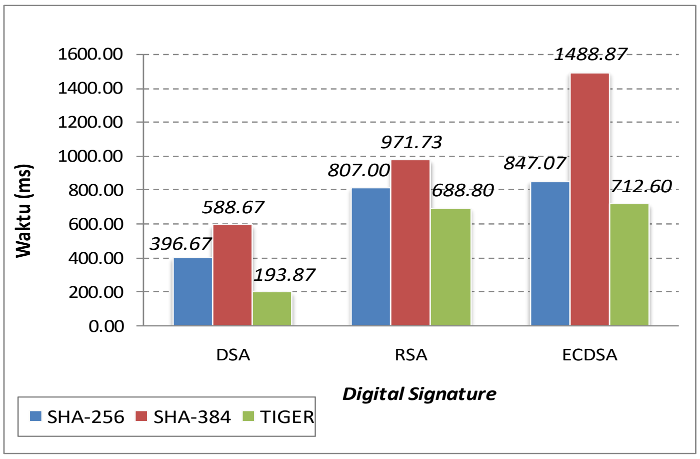

Figure 5. Comparison of the Processing Time for Verifying the DSA, RSA and ECDSA Algorithms

However, these results do not show definitively that the three algorithms are significantly different in terms of performance. A significant difference can also be seen in the use of the hash function. Therefore, an ANOVA test was carried out.

From the results of the ANOVA test during the verifying process, it can be seen that the resulting P (probability) values are all smaller than α (with α=0.05). These results conclude that the algorithm and hash function make a real difference in the verifying process.

Memory Usage Analysis for Signing Process and Verifying Process

The analysis at this stage is an analysis of the test results that compares the memory both during the signing and verifying processes.

Summary data on memory usage in the signing process is presented in Table 3. To make it clearer, the comparison results for signing memory usage are presented in the form of a graph in Figure 7. From these results, it is clear that the differences between the three are obvious.

Table 3. Memory Usage Summary Data in the Signing Process

Fungsi Hash	Memori *Signing* (KB)			Rata-rata Baris (KB)
	DSA	RSA	ECDSA	
SHA-256	54,169.87	1,976.00	2,969.33	19,705.07
SHA-384	54,193.87	1,976.00	2,969.60	19,735.20
TIGER	54,206.40	1,976.00	2,973.60	19,718.67
Rata-rata Kolom	54,190.04	1,998.04	2,970.84	

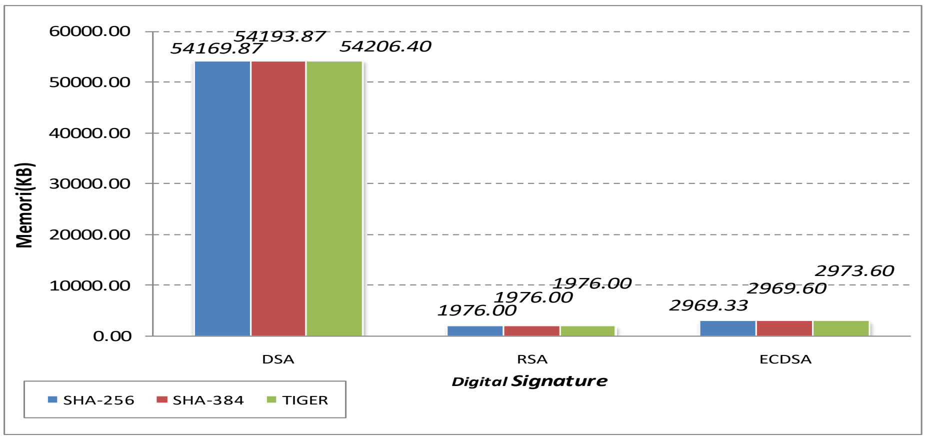

Figure 6. Comparison of Memory Usage for DSA, RSA and ECDSA Signing Processes

From the results of identification of memory usage for the signing process, it can be concluded that the digital signature algorithm is very influential. If you pay attention to the hash functions in each algorithm, the results are not much different, but if you look at it based on the algorithm, there are differences in the amount of memory required in large quantities for the signing process.

Verifying Process Memory Usage Analysis

Summary data on memory usage in the verifying process is presented in Table 4. To make it clearer, the comparison results for signing memory usage are shown in the form of a bar graph in Figure 8. From these results, it is clear that the differences between the three are obvious.

Table 4. Memory Usage Summary Data in the Verifying Process

Fungsi Hash	Memori *Verifying* (KB)			Rata-rata Baris (KB)
	DSA	RSA	ECDSA	
SHA-256	53,476.00	1,980.00	2,300.00	19,252.00
SHA-384	53,480.00	1,980.00	2,296.00	19,252.00
TIGER	53,484.00	1,979.73	2,296.00	19,253.24
Rata-rata Kolom	53,480.00	1,979.91	2,297.33	

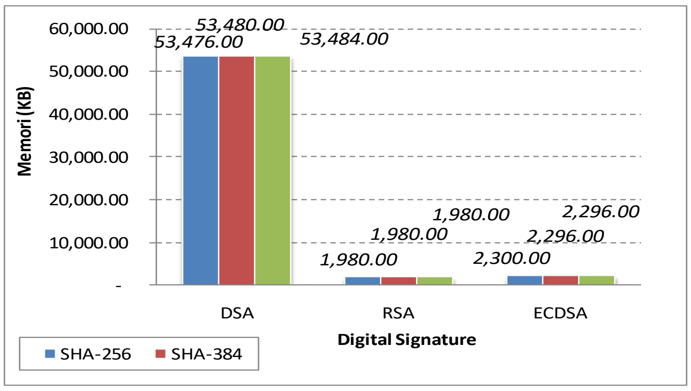

Figure 7. Comparison of Memory Usage for Verifying DSA, RSA and ECDSA Processes

From the results of memory identification for the verification process, it can be concluded that the digital signature algorithm is very influential. If you pay attention to the hash functions in each algorithm, the results are not much different, but if you look at it based on the algorithm, there is a large difference in the amount of memory required for the verifying process.

CONCLUSIONS

In this study a digital signature application has been designed using hash functions, namely SHA-256, SHA-384 and Tiger with DSS, namely DSA, RSA, and ECDSA using a C++ class library for cryptographic schemes. The application can receive input in the form of files with text, executable and image formats.

From the results of testing using statistical tests it is known that to identify the performance during the signing and verifying processes that have significant differences are the algorithm factors (due to the way or method of distributing hash values for algorithmic signature files and cryptosystem application development tools for digital signatures), the hash function factor (due to the way the hash function is distributed which has differences in design, block input capacity and number of rounds) as well as the interaction between algorithm factors and hash function factors. In various file sizes the combination of the DSA algorithm with the Tiger hash function is the best combination because it performs the fastest signing and verifying processes. In contrast, the combination of ECDSA with the SHA-384 hash function in the signing/verifying process is the worst combination because it takes the longest signing and verifying processes. The larger the file size, the fewer treatments that have similarities, and vice versa.

To identify the use of signing and verifying memory which has a significant difference is the digital signature algorithm which is affected by file size, while the hash function has no effect or has no significant difference. In DSA that does not apply the filter concept, the larger the file size, the greater the amount of memory needed, the opposite applies to RSA and ECDSA which apply the filter concept.

In future research in order to develop an understanding of cryptosystem applications, it is necessary to identify in the analysis the generation of a key pair and the use of a development environment other than the Crypto++ Library, if possible a comparison between the two is made. In addition, research can also be carried out that does not utilize the filter concept which is an advantage of the Crypto++ Library application for RSA and ECDSA, so that other differences in patterns can be identified in terms of processing time and memory usage.

REFERENCES

Anonim. Manual Crypto++ Library 5.6.0 API Reference. http://www.cryptopp.com, [11 Mei 2009].

Ismaliansyah MK. 2006. Kriptanalisis Pada Tiger. http://www.informatika.org/~rinaldi/Kriptografi/2006-2007/Makalah2/Makalah-057.pdf, [25 Februari 2011].

Menezes A., Van Oorschot P., & Vanstone S. 1996. Hanbook of Applied Cryptography. CRC Press Inc. www. cacr. math. uwaterloo. ca/hac. [15 Juni 2009].

Munir R. 2006. Kriptografi. Bandung: Informatika.

Rasyid MF. 2007. Tanda Tangan Digital Majemuk dengan Kunci Publik Tunggal dengan Algoritma RSA dan El Gamal. http://www.informatika.org/~rinaldi/Kriptografi/2007-2008/Makalah2/MakalahIF5054-2007-B-039.pdf, [September 2009].

Setiawan A. 2009. Percobaan Faktorial. http://smartstat.wordpress.com/2009/12/23/slide-rancangan-faktorial, [30 September 2010]

Stalling W. 2003. Cryptography and Network Security Principles and Practice. Third Edition. New Jersey: Pearson Education.

Walpole RE. 1990. Pengantar Statistika. Sumantri B, penerjemah. Jakarta. PT. Gramedia. Terjemahan dari: Introduction to Statistics.

Triwinarko A. 2004. Elliptic Curve Digital Signature Algorithm (ECDSA). http://www.infomatika.org/~rinaldi/TA/Makalah_TA%20Andy%20T.pdf, [September 2009].

Triyanto. 2009. Pengenalan Minitab. http://trie.staff.fkip.uns.ac.id/files/2010/03/MINITAB-14.pdf, [07 Januari 2011].