

**Computation Complexity Improvement for Digital  
Signature Algorithm**

**تحسين تعقيد الحسابات لخوارزمية التوقيع الرقمي**

**Prepared by**

**Mohammed Mustafa Rifaat**

**Supervisor by**

**Prof. Hamza Abbass. Al-Sewadi**

**Thesis Submitted In Partial Fulfillment of the Requirements  
for the Degree of Master of Computer Science**

**Department of Computer Science**

**Faculty of Information Technology**

**Middle East University**

**May, 2017**

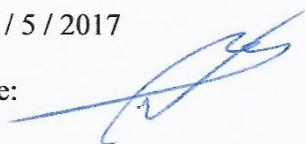
## Authorization Statement

I, Mohammed Mustafa Rifaat, authorize the Middle East University to provide hard copies or soft copies of my Thesis to libraries, institutions or individuals upon their request.

Name: Mohammed Mustafa Rifaat

Date: 31 / 5 / 2017

Signature:

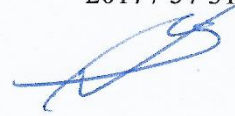


### إقرار تفويض

انا محمد مصطفى رفعت، افوض جامعة الشرق الاوسط بتزويد نسخ من رسالتي ورقياً او الكترونياً للمكتبات او المنظمات او الهيئات و المؤسسات المعنية بالابحاث و الدراسات العلمية عند طلبها.

الاسم: محمد مصطفى رفعت

التاريخ: 2017 / 5 / 31

التوقيع: 

**Middle East University****Examination Committee Decision**

This is to certify that the thesis entitled "**Computation Complexity Improvement for Digital Signature Algorithm**" was successfully defended and approved on 31/ 5 / 2017

**Examination Committee Members****Signature**

**Prof. Hamza Abbass. Al-Sewadi** (Supervisor and Chairman)  
**Professor, Department of Computer Information Systems**  
**Middle East University (MEU)**

*Hamza A-Al-Sewadi*  
*10-6-2017*

**Dr. Mudhafar Al-Jarrah** (Internal Member)  
**Assistant Professor, Department of Computer Science**  
**Middle East University (MEU)**

*Mudhafar Al-Jarrah*

**Prof. Mohammad Mustafa Sulaiman Al-Haj Hassan** (External member)  
**Professor, Department of Computer Science**  
**Zarqa University (ZU)**

*M. M. S. Al-Haj Hassan*

## **Acknowledgments**

Praise to Allah Almighty for everything and for permitting me complete the study journey by complete this Thesis and writing this part of it. I want to thank my family for their support and pray for me. Me and my family thanks Prof. Hamza Abbass for his patient, understanding, supporting and guiding me as teacher and father to study and complete this thesis, and the thanks continue to Prof. Ali Makki for all his support and notes, which helped me a lot in this thesis. Big thanks for my friends who support me in this journey and for my mate along this study journey Ayssar Alssadi who we lived together, days of tiredness, sadness and happiness, i will remember them for long my life, thank you for supporting me during along this study hard way. I also want to thank "E H" for supporting me along this study journey. Thank you all for everything.

Finally,I want to thank my teachers and members of IT faculty in Middle East University.

Mohammed Mustafa

The Researcher

## **Dedication**

To my family and the soul of my father, my friends, close persons, and all who love me and misses me during my study journey. I dedicate my work.

Mohammed Mustafa

## Table of Contents

Cover Page .....	I
Authorization Statement.....	II
إقرار تفويض .....	II
Examination Committee Decision .....	IV
Acknowledgments .....	V
Dedication .....	VI
Table of Contents .....	VII
List of Tables.....	XII
List of Figures .....	XII
Table of Abbreviations.....	XIII
Abstract .....	XV
الملخص .....	XVII
Chapter One: Introduction.....	1
1.1 Introduction .....	2
1.2 Definitions .....	3
1.3 Cryptography.....	5
1.4 Problem Statement .....	7
1.5 Questions of the Study .....	7
1.6 Objectives of the Study .....	8
1.7 Motivation .....	9
1.8 Scope and limitations .....	9
1.9 Thesis Organization.....	10
Chapter Two: Theoretical Background and Literature Review.....	11
2.1 Introduction .....	12
2.2 Mathematical Preliminaries.....	12

2.2.1 Modular Arithmetic .....	12
2.2.2. Divisors .....	13
2.2.3. Exponentiation in Modular Arithmetic .....	14
2.2.3.1 Algorithm (Fast Exponential).....	14
2.2.4. The Multiplicative Inverse in $F(p)$ .....	15
2.2.4.1 Algorithm (Extended Euclidean algorithm).....	15
2.2.5 Discrete Logarithms in Cryptography .....	17
2.3 Theoretical Background .....	18
2.4 Digital Signature Techniques .....	20
2.4.1 Diffie-Hellman (D-H) .....	21
2.4.2 RSA algorithm.....	22
–Key generation .....	22
–RSA Signing .....	22
–RSA Verify .....	22
2.4.3 ElGamal.....	23
–ElGamal Keys Generation .....	23
–ElGamal Signing process .....	24
–ElGamal Verification process .....	24
2.4.4 Schnorr Digital Signature.....	24
–Schnorr Signing process.....	24
–Schnorr Verification process.....	25
2.4.5 Digital Signature Algorithm (DSA) .....	25
–DSA Key generation .....	25
–DSA Signature Generation Process .....	26
–DSA Verification process .....	26
2.4.6 Short Comparison.....	27



2.4.7 The DSA Variants .....	27
2.4.7.1 GOST Digital Signature .....	28
–GOST Signature process: .....	28
–GOST Verification process: .....	28
2.4.7.2 Yen-Laih Digital Signature .....	29
–Yen-Laih Signature Generating Process .....	29
–Yen-Laih Verification Process .....	29
2.4.7.3 McCurley Digital Signature .....	29
–McCurley Signature Generating Process .....	30
–McCurley Verification Process .....	30
2.4.7.4 Variant of DSA.....	30
2.5 Message Digest .....	31
2.6 Blind Digital Signature.....	34
2.7 Group Digital Signature .....	34
2.8 Group Blind Digital Signature .....	35
2.9 Review of Related Literature .....	35
Chapter Three: The Proposed Modified DSA.....	39
3.1 Introduction .....	39
3.2 Investigation of DSA algorithms methodology .....	39
3.3 The proposed Modified DSA (M.DSA).....	43
3.4 The modified M.DSA proof .....	45
3.5 Digital Signature Algorithm.....	47
3.6 Security Concerns of M.DSA.....	53
3.7 Key Strength.....	54

Chapter Four: Implementation and Results.....	55
4.1 Introduction .....	56
4.2 Digital Signature Implementation .....	56
4.3 Digital Signature Comparison.....	59
4.4 Speed Gain .....	60
4.4.1 Signature Speed Gained .....	60
4.4.2 Verify Speed Gained .....	61
4.5 DSA Variants Implementation .....	62
4.6 Algorithms Signature and Verify complexity .....	65
4.6.1 DSA signature complexity: .....	65
4.6.2 DSA Verify complexity .....	66
4.6.3 M.DSA signature complexity.....	67
4.6.4 M.DSA Verification complexity .....	68
4.6.5 GOST Signature complexity .....	69
4.6.6 GOST Verify complexity .....	69
4.6.7 Yen-Laih Signature Complexity .....	70
4.6.8 Yen-Laih Verification Complexity .....	71
4.6.9 McCurley Signature Complexity .....	72
4.6.10 McCurley Verification Complexity .....	72
4.6.11 Variant of DSA Signature Complexity .....	73
4.6.12 Variant of DSA Verification Complexity .....	74
4.7 Number of Operations in Algorithm .....	76
4.7.1 Signature Generating Process.....	76
4.7.2 Verification Process .....	77

Chapter Five: Conclusion and Future Work .....	79
5.1 Conclusion .....	80
5.2 Future Work .....	81
References .....	82
Appendix A .....	85

### List of Tables

<b>Chapter Number Table Number</b>	<b>Contents</b>	<b>Page</b>
Table (2.1)	Result of the Fast Exponentiation Algorithm	(15)
Table (2.2)	Finding the Multiplicative Inverse	(17)
Table (2.3)	Comparison of Listed Algorithms in Chapter Two	(27)
Table (4.1)	Parameters of Algorithms with Primes Length of 100 Decimal Digits	(57)
Table (4.2)	Signing and Verification Time Calculation for DSA, with p, q and g of Length of 100	(57)
Table (4.3)	Digit, 10 Values of k For Each x Value Signing and Verification Time Calculation for M.DSA, with p, q and g of Length of 100 Digit, 10 Values of k for Each x Value	(58)
Table (4.4)	An Average of ((900 Signature and 900 Verify) * 15 Round) Time in ms Compare for Several Algorithms with M.DSA	(59)
Table(4.5)	Signature Speed Gained For M.DSA Over Other Algorithms	(61)
Table (4.6)	Verify Speed Gained of M.DSA Over Other Algorithms	(61)
Table (4.7)	VAR-DSA Algorithm	(62)
Table (4.8)	GOST Algorithm	(63)
Table (4.9)	McCurley Algorithm	(64)
Table (4.10)	Yen-Laih Algorithm	(64)
Table(4.11)	Summary of BigO Notation	(75)
Table (4.12)	Algorithm Number of Arithmetic Operations in Signature	(76)
Table (4.13)	Algorithm Number of Arithmetic Operations in Verification	.(77)

### List of Figures

<b>Chapter Number Figures Number</b>	<b>Contents</b>	<b>Page</b>
<b>Figure(1)</b>	Basic Concept of digital signature	<b>4</b>
<b>Figure(2)</b>	Digital Signature Model	<b>19</b>
<b>Figure(3)</b>	Block diagram of signing and verification processes of DSA	<b>26</b>
<b>Figure(4)</b>	Hash Function	<b>33</b>
<b>Figure(5)</b>	Flowchart for measuring the time complexity for the algorithm.	<b>41</b>
<b>Figure(6)</b>	the steps followed in the modified algorithm	<b>45</b>
<b>Figure(7)</b>	Comparison of total time of Variants of DSA with M.DSA	<b>60</b>
<b>Figure(8)</b>	Comparison of Variants of DSA BigO	<b>75</b>
<b>Figure(9)</b>	Comparison of Verify and Signature Operations	<b>78</b>

### Table of Abbreviations

Abbreviations	Meaning
DSA	Digital Signature Algorithm
MAC	Media Access Control
IP	Internet Protocol
RSA	Rivest, Shamir, and Adelman
DSS	Digital Signature Standard
FIPS	Federal Information Processing Standards
DH	Diffie Hellman
SHA	Secure Hash Algorithm
MD5	Message Digest 5
VSC	Vehicle Safety Communication
EC	Elliptic Curve
NIST	National Institute of Standards and Technology
$K_r$	Private Key
$K_p$	Public Key

# **Computation Complexity Improvement for Digital Signature Algorithm**

**Prepared by  
Mohammed Mustafa Rifaat**

**Supervisor  
Prof. Hamza A. Al-Sewadi**

## **Abstract**

Digital signature is an authentic method of the received data/messages in the digital world. It is considered to be equivalent to hand written signature which provides the authenticity of the signed papers. Digital signature algorithm (DSA) adopted for the first time at 1991 by National Institute of Standards and Technology (NIST) for use in their digital signature standard. DSA uses Discrete Logarithm Problem (DLP) to generate signature for signed messages. Many variants of DSA appeared and has been suggested in order to improve the performance, such as Yen-Laih, GOST, McCurley, and other algorithms.

In this thesis a modified version of DSA (referred to as M.DSA) is proposed with the aim of enhancing the time complexity measurement, i.e. looking for faster implementation. Mathematical proof of this modified version is also included.

The work starts with an investigation for the DSA and four of its variants in order to examine the effect of the digital signature parameters variations on their performance. The average time for signing and verification for the original DSA, modified DSA, and all the other variants are computed for various parameter lengths of private keys,

randomly generated keys, and messages, then comparison between all results was conducted. Experimental computations have shown clearly that M.DSA has better validation time as compared with others. Also, the overall time complexity was impressive, and the speed gain about two times the original DSA overall time. Hence it is recommended that this modified version of DSA would be used for applications that require fast verification time.

**keywords:** Digital Signature, Authenticity, Digital Signature Algorithm (DSA), Discrete Logarithm Problem (DLP), Variants of DSA, Signature, Verification, Private Key



# تحسين تعقيد الحسابات لخوارزمية التوقيع الرقمي

إعداد

محمد مصطفى رفعت

إشراف

الأستاذ الدكتور حمزة عباس السوادي

الملخص

التوقيع الرقمي هو طريقة للتأكد من وثوقية مصدر البيانات او الرسائل المستلمة ضمن

مجال العالم الرقمي حيث يعتبر مكافئ للتوقيع المكتوب باليد و الذي يُعبّر عن مصدر هذه

المستندات وثوقيتها، إن خوارزمية التوقيع الرقمي DSA اعتمدت من قبل المعهد الوطني للمقاييس

التكنولوجيا لأول مره عام 1991 لتستخدم في توقيع بياناته، خوارزمية التوقيع الرقمي تستخدم

اللوغاريتمات المنفصلة لتوليد التواقيع الرقمية للرسائل المُوقعة، ظهرت العديد من الخوارزميات

الشبيهة لخوارزمية التوقيع الرقمي، إضافة إلى عدة مُحاولات لتحسين أداء هذه الخوارزمية، منها

Yen-Liah, GOST, McCurley و خوارزميات اخرى.

تحتوي هذه الاطروحة على خوارزمية معدله من خوارزمية التوقيع الرقمي ( أطلق عليها

اسم M.DSA )تهدف إلى تحسين تعقيد خوارزمية التوقيع الرقمي (DSA) لتنفيذ الخوارزمية بشكل

اسرع، و تحتوي هذه الرسالة على الإثبات الرياضي لهذه الخوارزمية المعدلة.

يبدأ العمل بإجراء اختبار لخوارزمية التوقيع الرقمي الاصلية و كذلك اربع إصدارات مختلفة

منها لإختبار تأثير مُعاملات التوقيع الرقمي على إداء هذه الخوارزميات. تم حساب و مقارنة معدل

وقت التوقيع و التحقق منه لاطوال مختلفة لمعاملات خوارزمية التوقيع الرقمي الاصلية، و النموذج

المُقترح و كل الإصدارات التي تم ذكرها مسبقاً لمعامل المفتاح الخاص و الرقم العشوائي و كانت

نتيجة المقارنة تشير إلى تفوق في وقت التحقق من التوقيع و كذلك التعقيد العام للخوارزمية المعدلة على بقية الخوارزميات حيث كانت اسرع بحوالي مرتين من الخوارزمية الاصلية بشكل عام. ومن هنا يوصى باستخدام الإصدار المُقترح من الخوارزمية في التطبيقات التي تحتاج سرعة في حساب وقت التحقق من التوقيع.

**الكلمات المفتاحية :** التوقيع الرقمي ، وثوقية، خوارزمية التوقيع الرقمي ، مشكلة اللوغاريتمات المنفصلة، الخوارزميات الشبيهة لخوارزمية التوقيع الرقمي، توقيع، تحقق، المفتاح الخاص.

# **Chapter One**

## **Introduction**

# **Chapter One**

## **Introduction**

### **1.1 Introduction**

Digital documents and messages are stored and widely exchanged over the internet and communication networks nowadays. Most of these messages are personal or governmental with confidential or sensitive information, and would lead to decisions making, money transfer, election voting, or any other important consequences. Accordingly, the sender must be a trusted person and the receiver must ensure that the received message was not altered by any intruders during its transfer. However, knowing that the received message is not altered and was sent from a trusted person requires a robust method to ensure its authenticity. Digital Signature is one of the best ways used so far to authenticate the digital sensitive data and messages over the networks or the internet. So many digital signature techniques have been developed and implemented in various applications. As in hand-written signature has a unique way for every person that gives identity of the person who signed the document, in the digital world the digital signature also has unique values and parameters that must change for each message to ensure the provenance so both of them are used for signing messages, documents or contracts but keeping the physical difference between them in consideration. The required time for signing and verification of digital document is crucial. In this project, it is planned to investigate the widely used digital signature algorithms in terms of security and time complexity measurement for both signing and verification processes with the aim of getting more efficient and faster scheme. The improvement in signing or verification time is sought through the use of less computation steps while care must be taken for the security factors. Also

implementation of such improved schemes would be considered for some relevant applications.

## 1.2 Definitions

It is important at the beginning to clarify the exact meaning of the common terms used in the field of the study. Computer data and information security term includes cryptography, data hiding, and digital signature, hence these terms will be briefly defined in the following (Stallings, 2013).

1. **Computer Security:** is the protection of the systems and their information by achieving the basic security concepts requirements for them as follows:
  - Integrity: guard the information against modifications.
  - Availability: ensuring reliable access and use of information.
  - Confidentiality: protect information from unauthorized access, in other words protecting personal privacy.
2. **Cryptography:** Study of methods and techniques to secure messages and data during transmission by converting a clear and intelligible text (referred to as plain text) into an unintelligible form (referred to as cipher text)
3. **Data hiding:** Study of methods and techniques for embedding the information into another multimedia. It comes in two types; steganography and watermarking, where
  - **Steganography:** hiding information in media to keep the data hidden from public, for example computer, files like images videos and other types of media files. It is primarily used for cryptography.
  - **Watermarking:** embedding information such as a signature or a logo into a multimedia in media in order to protect its ownership or copyright.

4. **Digital Signature:** An equivalent mathematical form of hand written signature. In digital world, this method is used to authenticate a received message and ensure that the sender is the person who claims to be (Zhang, Li, & Song, 2011).

This research work is concerned with the improvement of digital signature and therefore, more explanation will be included here. In Modern systems the digital signature is very important because it provides: (Engström, 2016):

- **Confidentiality:** If the message is signed and encrypted, no third person can decrypt it if he doesn't have the keys.
- **Authentication:** Digital signature provides verification of the signer.
- **Integrity:** It provides a way to protect the message from being altered.
- **Non - repudiation:** The signer cannot deny the signature because no one else has the signing key.

Generally the digital signature scheme either signs the message or its hash using a signing algorithm, then the receiver applies a verifying algorithm to the message or its hash in order to produce an information that tells the correct validity of the signer's identity or not as shown in Figure(1).

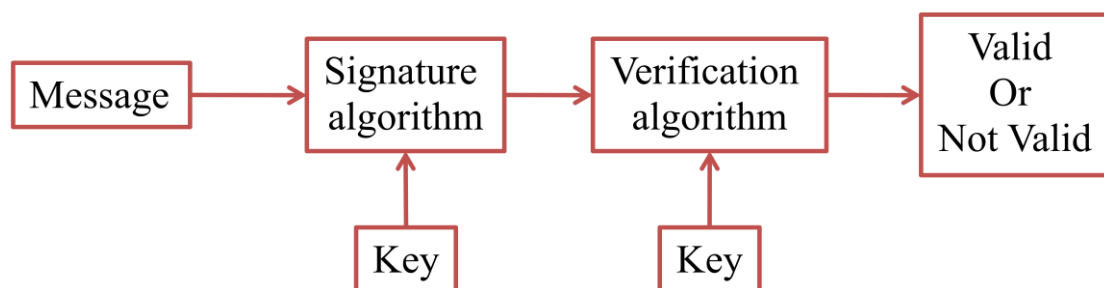


Figure (1) Basic Concept of digital signature

There are many digital signature algorithm such as Rivest, Shamir, and Adelman (RSA), Schnorr algorithm, ElGamal algorithm, Digital Signature Algorithm (DSA), and Elliptic Curve Digital Signature Algorithm (ECDSA), and many others. The most common algorithm will be explained in details in chapter two. These algorithms vary in the way of mathematical generating signature and verification which affect the speed and performance of the algorithm.

### 1.3 Cryptography

Building a system of rules allow two or more parties to communicate with each other, and at the same time prevent outside entities, parties or the public (adversaries), as they called in cryptography, from reading the communication private messages.

Cryptography uses process called *encryption* to transform the plain text or usable data into form which can't be useful for any person except the authorized persons. This process is done by using a shared secret key between the communication sides, then recover the original form of the data at the end side by using the key again, this process is called *decryption*. Modern Cryptography is based on mathematic and computation hardness. It can secure information by providing data confidentiality, data integrity, authentication and non-repudiation. Therefore, it is used in various applications like e-commerce and ATM.

At the 60's, the communications and computer systems came with the ability to store and protect the private information in the digital form using special mechanisms like Data Encryption Standard (DES) which was the standard for secure the e-commerce. At late of 70's a big change in cryptography came with the concept of public key cryptography by Diffie and Hellman in their paper "New Directions in

Cryptography", which also gives a new brilliant way for key exchange. Their method depends on the complexity of the discrete logarithm problems, and the idea inspired the cryptography community. One year later, Rivest, Shamir and Adleman came with practical method public key encryption. The method also used in digital signature, and based on factoring large integers. At 1985, another method for public key was found by ElGamal based on the described logarithm problem. In 1991, an international standard of digital signature was adopted and based on RSA. Three years later, U.S Government embraced the Digital Signature Standard which is based on ElGamal method.

Cryptography serves the information security requirements using mathematical techniques, and its strength is measured by time and resources it would require to verify or recover the information. Keeping the information safe from the unauthorized parties ensures the privacy of the transmitted information even over unsecure communication channels. A secured information using cryptography methods can protect the connections parties from being received faked information or altered information by unauthorized person from the outside of trusted parties that communicate by providing data integrity. Communication sides can be identified if the cryptography methods are used in communication because of their parameters. Signing the transmitted message is possible with cryptography using digital signature method, and performing attack on a strong cryptography system may need the power of billions of computers and all the available time without getting result of decipher or get the required value (Network Associates, Inc). But no one can approve that even the most powerful cryptosystem can hold up against tomorrows computing power,



## **1.4 Problem Statement**

Digital signature is a method to fight cybercrimes like identity theft in internet. It is equivalent to ink-paper or the hand written signature which is used to authenticate the documents or bills etc., in addition to proving identity as the handwritten signature the digital signature provide non-deny signature, i.e. the sender cannot deny his or her signature because of the unique way for the sign method and parameters like hash value of the message, random number for each message and the private number. All these factors make the sender can't deny his signature and provide authenticity of the sender as well. Therefore many methods were developed for generating digital signature, from these methods: Digital Signature Algorithm (DSA), Rivest, Shamir, and Adelman (RSA) and elliptic curve digital signature algorithm (ECDSA) and many other methods derived (Variants) from these methods and other methods. These methods vary in performance, required time to sign and verify, and complexity.

Since digital signature algorithm is used in various applications where long time for signing, verification, or both is not acceptable, therefore improving the execution speed of DSA is essential. The main problem investigated in this study is how to improve the execution speed for the digital signature, i.e. to reduce the signing and verification time and also keeping the high security of the DSA.

## **1.5 Questions of the Study**

This research is aimed to look for answers for the following questions:

1. How to test the effect for various factors on the DSA algorithm?
2. Will adding new factors to the algorithm have any effect on the signing and verifying time?

3. Does this addition of new factors affect the security performance?
4. How the achieved improvements in the calculation time will be reflected on the digital signature algorithm performance for practical applications if the random numbers generated using characters or letters instead of integers?
5. Is the length of private key an essential factor on the signing speed?
6. Does the length of message secret integer random affect the signing generating process speed?

## **1.6 Objectives of the Study**

The computations of digital signature algorithm generally rely mainly on the choice of large primes and use of discrete mathematics, involving multiplications and exponentiations of large numbers, and their security relies on the difficulty of analyzing and factoring of large numbers. Therefore, one of the possible ways to improve the time measurement, which is the goal of this study, is the choice of the parameters used and the way in which the mathematical processes are done.

The main objectives of this study are:

1. Investigate the effect of important factors on the time measurements for signing and verification of the standard digital signature algorithm, which is approved by the National Institute of Standards and Technology (NIST).
2. Investigate digital signature algorithm versions that claim enhancement in verification time.
3. Propose, design, and test a modified version of the digital signature algorithm which involves the same parameters as these for standard DSA algorithm but gives enhancement in the signature time and verification time or both.

## **1.7 Motivation**

1. Improvement of computation complexity by reducing the need for multiplication and exponentiation processes used in the DSA will certainly enhance the time needed for the signing and validation process, and thus increase the efficiency.
2. Some modification versions of digital signature algorithm have shorter signing time, while others have shorter verification time. Therefore, a worthy motivation to work on a modification to digital signature algorithm that improves both signing and verification time. Such improvement will be useful for sender and verifier for application where time is a crucial issue.

## **1.8 Scope and limitations**

Digital signature technique is considered essential for both data integrity and sender authentication, and hence it is badly required for digital documents and messages. Speed of signing and verification is a crucial factor for the digital signature due to time and efforts (or power requirement) for some applications. As digital signature normally relies on the difficulty of factoring large integers, hence, efficient algorithms are sought that can facilitate the required computational difficulty and the capability of using integers with convenient size. Besides, the lengthy mathematical operations, especially using iterative modular multiplication and exponentiation are time consuming and present another limitation on the time requirement. These are the major limitations on the digital signature algorithm involved. On the other hand, some offline calculation steps help to speed up the operations. Moreover, the search for effective algorithm with fewer mathematical operations will help to soften this problem. Also the use of Personal computer for this thesis research work presents another

limitation on the size of the parameters used. However this research work represents a prototype scheme.

## **1.9 Thesis Organization**

This thesis is organized in the following way.

Chapter One: Gave a brief introduction to the problem of digital signature. After the definition of data security and digital signature, the problem to be investigated is stated and the objectives and motivation are included.

Chapter Two: Include theoretical background related to the digital signature computations, applications of digital signature, digital signature techniques and a review of related literatures.

Chapter Three: Introduces the methodology used in this thesis, an investigation of the digital signature algorithm, and description for the DSA algorithm with mathematical proof, and an example, the proposed algorithm and its mathematical proof with example of digital signature using the proposed algorithm. A key strength and security concerns of the proposed algorithm are also included.

Chapter Four: Covers implementation of the proposed algorithm and comparing it with the original DSA and some other similar algorithms of digital signature, which shows improvement in complexity and time of verification.

Chapter Five: Concludes this thesis, and gives suggestions for future work in digital signature.

# **Chapter Two**

## **Theoretical Background and Literature Review**

## Chapter One

### Theoretical Background and Literature Review

#### 2.1 Introduction

This chapter will give a brief definition and theoretical background for the digital signature processes together with outline of the widely used and relevant digital signature techniques. Then, literature review of related available works will be included.

#### 2.2 Mathematical Preliminaries

In this section mathematical subjects are used in cryptography will be introduced in brief to clear the ambiguity of some cryptography calculations, like the  $k^{-1}$  which is the multiplicative inverse operation, where  $k \cdot k^{-1} = 1 \bmod p$ , not as in normal mathematics  $k^{-1} = \frac{1}{k}$ . For the clarity of calculations, some algorithms and methods have to be included in this thesis.

##### 2.2.1 Modular Arithmetic

Given any positive integer  $n$  and any nonnegative integer  $a$ , then  $r = a \bmod n$  can be computed if  $a$  is divided by  $n$  to get an integer quotient  $q$  and an integer remainder  $r$  illustrated in equation (1).

$$a = qn + r, \text{ where, } 0 \leq r < n, \text{ and } q = \left\lfloor \frac{a}{n} \right\rfloor \dots\dots\dots(1)$$

Where  $\lfloor a \rfloor$  is the largest integer less than or equal to  $a$ .

Example (1):- let  $a=37$  and  $n=17$

$$r \equiv 37 \bmod 17 \equiv 3$$

Where,  $a = qn + r$

$$37 = 2 \cdot 17 + 3$$

Therefore,  $q = 2$  and  $r = 3$ .

Example (2):- For negative integer, let  $a = -37$  and  $n = 17$

$$r \equiv -37 \bmod 17 \equiv 14$$

Where,  $a = qn + r$

$$-37 = -3 \cdot 17 + 14$$

Therefore,  $q = -3, r = 14$ .

### 2.2.2. Divisors

$b$  is said to be nonzero divisor of  $a$  if  $a = mb$  for some  $m$ , where  $a$ ,  $b$ , and  $m$  are integers. That is,  $b$  divides  $a$  if there is no remainder on division. The notation commonly used to mean  $b$  divides  $a$  is  $b|a$ .

The positive divisors of 15 are 1, 3, 5, and 15.

The following properties of divisor hold:

If  $a|1$ , then  $a = \pm 1$ .

If  $a|b$  and  $b|a$ , then  $a = \pm b$ .

Any  $b \neq 0$  divides 0.

### 2.2.3. Exponentiation in Modular Arithmetic

The Digital Signature operation (signature and verification) in DSA calculation involves power of an integer to an integer, mod  $p$ . The intermediate values computation are reduced by modular  $p$  in the exponentiation done over the integers according the property of modular arithmetic

The fast power (or fast exponential) algorithm for computing  $a^b \bmod p$ , is one of the most common algorithms applied in various techniques of cryptography, which can compute  $a^b \bmod p$  in  $\log(n)$  time complexity. The fast power algorithm is given in the following:

#### 2.2.3.1 Algorithm (Fast Exponential)

Input: an integer  $a$  as base, an integer  $b$  as exponential, and an integer  $p$  as modular

Output: result  $c = a^b \bmod p$ .

Step 1: Convert  $b$  into binary  $b_{n-1}, b_{n-2}, \dots, b_2, b_1, b_0$ .

Step 2: Set  $c=1$

Step3: for  $i = n-1$  down to 0

3.1:  $c = c * c \bmod p$

3.2: if  $b_i = 1$  then

$c = c * a \bmod p$



Step 4: output c

Example:-

Table (2.1) Result of the Fast Exponentiation Algorithm

Result of the Fast Exponentiation Algorithm for $a^b \bmod p$ , where $a = 3$ , $b = 83_{10} = 1010011_2$ , $p = 97$							
i	6	5	4	3	2	1	0
$b_i$	1	0	1	0	0	1	1
c	3	9	49	73	91	11	72

#### 2.2.4. The Multiplicative Inverse in $F(p)$

The multiplicative inverse of an element (or integer) in  $F(p)$  for small values of  $p$  can be calculated easily. But, for large numbers  $p$ , it is not practical.

If  $\gcd(p, a) = 1$ , then  $a$  has a multiplicative inverse modulo  $p$ . That is, for positive integer  $a < p$ , there exists  $a^{-1}$  such that  $aa^{-1} = 1 \bmod p$ . The extended Euclidean algorithm can return the multiplicative inverse of  $a$ , in addition to finding  $\gcd(p, a)$ . If  $\gcd(p, a)$  is 1, then  $a$  is said to be relatively prime to  $p$ .

The extended Euclidean algorithm is illustrated in the next page.

##### 2.2.4.1 Algorithm (Extended Euclidean algorithm).

Input:  $a, p \in \mathbb{N}$  where  $p \geq a$ , and  $a > 0$ .

Output: numbers  $b_2 \in \mathbb{Z}$  such that  $a \cdot b_2 = 1 \bmod p$ .

Step 1: if  $\gcd(a, p) \neq 1$  then

Exit, there is no Inverse

Step 2: Set  $a_1 = 1$ ,  $a_2 = 0$ ,  $a_3 = p$

$$b_1 = 0, b_2 = 1, b_3 = a$$

Step 3: While ( $b_3 \neq 1$  and  $b_3 \neq 0$ )

$$3.1: \quad q = \left\lfloor \frac{a_3}{b_3} \right\rfloor$$

$$3.2; \quad t_1 = a_1 - b_1 * q$$

$$t_2 = a_2 - b_2 * q$$

$$t_3 = a_3 - b_3 * q$$

$$3.3: \quad a_1 = b_1$$

$$a_2 = b_2$$

$$a_3 = b_3$$

$$3.4: \quad b_1 = t_1$$

$$b_2 = t_2$$

$$b_3 = t_3$$

Step 4: if  $b_3 = 1$  then

4.1 if  $b_2 < 0$  then

$$b_2 = b_2 + p$$

Step 5: Output  $b_2$  as  $a^{-1} \bmod p$

Table 2.2 shows an example of the execution of the algorithm. It shows that  $\gcd(29, 5) = 1$  and that the multiplicative inverse of 5 is 6 mod 29; that is,  $5 \times 6 \equiv 1 \bmod 29$

Table (2.2) Finding the Multiplicative Inverse

Finding the Multiplicative Inverse of 5 in $F(29)$						
q	a1	a2	a3	b1	b2	b3
	1	0	29	0	1	5
5	0	1	5	1	-5	4
1	1	-5	4	-1	6	1

### 2.2.5 Discrete Logarithms in Cryptography

The public-key cryptography technique such as RSA is designed according to the computational difficulty of Integer Factoring Problem solving into its large primes. The ElGamal cryptosystem, Diffie-Hellman key exchange and some of public-key cryptosystems are designed according to another computationally difficult number theory problem, which is Discrete Logarithms Problem (DLP). The Digital Signature Standard and Algorithms are designed according to solving the computing discrete logarithms.

## 2.3 Theoretical Background

Digital Signature is a mathematical way to sign digital documents in the digital world. It is a technique that insures both the integrity of the received digital data and the authenticity of communicating body or person (Bruce, S., 1996). Therefore it is an electronic signature analogous to the hand written signature. Sending documents over the internet is fraught with dangers of altering, changing, masking, denial, identity theft and other types of hazards. The signed data can't be denied by the sender because of the unique way and parts it contains in the signing process (Non-repudiation), which provides authenticity. It can be used even over non-secure communication channels as it provides the sender's identity and ensures that the document (data) has not been changed or altered during transmission. Digital signature is required in so many applications in everyday activities, such as money transfer, identity approval, personal authenticity, non-repudiation, e-government systems (Zhu and Xiao, 2008), it is also used in copyright and ownership protection of various documents, such as its use in PDF files of Adobe company (Chang, 2009).

Digital signature must have the followings requirements:

1. Dependent on the message.
2. Use information unique to sender in order to prevent both forgery and denial.
3. Relatively easy to sign any file and to verify the signature.
4. Computationally infeasible to forge genuine or fraudulent signature.
5. Practical to be saved locally.

Generally Digital Signature consists of the following three parts, see Figure (2) (Stallings, 2013).

1. Keys (public and private keys): the public key is generated and agreed upon by the communicating parties beforehand, while the private key is generated by the signer and may be changed at any time according to his will.
2. Digital signature generation algorithm.
3. Digital signature verification algorithm.

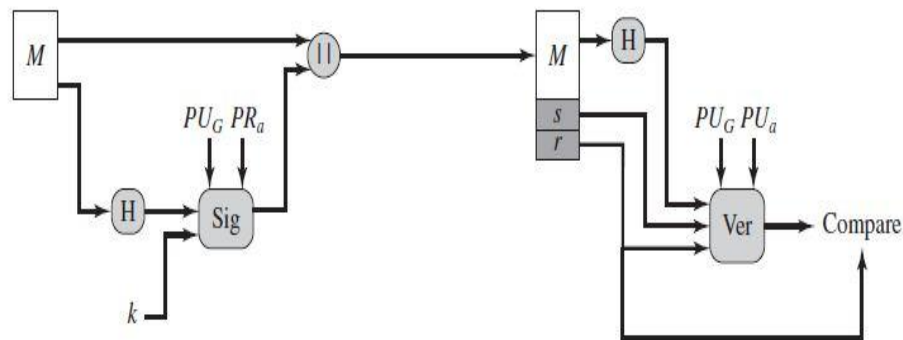


Figure (2) Basic Concept of digital signature

The signing and verification algorithms are agreed upon by the communicating parties.

To sign a message  $M$ , either the message or its hash is signed by the sender first using his own private key  $K_r$ , then the signed message is sent to the receiver with related parameter (if any), who will use the available public key  $K_p$  with the verification algorithm in order to verify the sender's identity. Digital signature algorithms benefit from the difficulty of calculating the Discrete Logarithms and large exponentiations in order to ensure security of keeping both communicating sides of being able to authenticate their message. Hence, this increase in the system computation complexity will increase the difficulty of breaking the security or recovering the signer's keys in order to forge his/her signature.

The following section outlines the widely known digital signature techniques that are relevant to the proposed work of this study.

## **2.4 Digital Signature Techniques**

Historically, public key implementation for digital signature started by two persons in 1976 (Whitfield Diffie and Martin E. Hellman at Stanford University in United State of America). They introduced a new way to exchange keys for more secure communication. They introduced a new method to generate and share cryptographic keys, that later had led to the invention of the algorithms, termed asymmetric encryption/decryption system algorithms as compared to the traditional symmetric system (Diffie and Hellman, 1976). Next year Ronald Linn Rivest, Adi Shamir, and Leonard Adleman created the RSA algorithm, (RSA is the name taken from their names first letters), using a new exchange way. The RSA algorithm was suitable for encryption/decryption as well as for digital signature (Rivest, Shamir and Adleman, 1978). However, it is more convenient for the encryption/decryption processes because it is slow for digital signature. Moreover, RSA algorithm turned out to be more useful for secret key distribution.

Later on, other digital signature algorithms were developed, namely El-Gamal algorithm (ElGamal, 1985), Schnorr algorithm (Schnorr, 1989), and the digital standard algorithm (DSA) by NIST 1991 with last revision at 2013 (Kerry, Gallagher, 2013). Few DSA variants were also developed. In the following, a brief description of these algorithms is included.

### 2.4.1 Diffie-Hellman (D-H)

Whitfield Diffie and Artin E. Hellman invented a practical protocol for the exchange of information for secure transmission over unsecured communications channels. The algorithm relies on the ease of calculating integer exponentiations and the difficulty of doing the inverse calculations to find the root numbers. The communication has to be secure in order to ensure the privacy, information and key exchange, and use them to secure the transmitted data which was one of the first problems in the field of cryptography. The algorithm steps are as follows:

1. The two communicating sides first agree upon two large prime integers'  $p$  and  $g$ , and keep them publicly.
2. Each side chooses a private key ( $k$ ), and calculates  $r = (g^k \bmod p)$  and sends the result publicly to the other side.
3. Now each side uses the received  $r$ , and calculates the shared secret key  $k_s$  by  $k_s = k^r \bmod p$ . This key  $k_s$  will be used as the secret key for encryption and decryption of the message or its hash value.

No one else except the sender and receiver knows this key,  $k_s$ , therefore encrypting any message with this key can be used as the proof of authenticity.

The D-H algorithm weakness is its vulnerability to man-in-the middle attack.

### 2.4.2 RSA algorithm

The security of this algorithm relies on factoring large numbers, i.e. the hardness of finding its prime roots number from a result of numbers with power. It consists of the following parameters,  $p, q, e, \phi, N$  &  $d$ . which are defined and used as in the following steps (Fu and Zhu, 2008), (Thangavel, 2014), (Jaju and Chowhan, 2015):

– **Key generation**

1. Choose two large positive prime integer numbers  $p$  and  $q$  (similar bit-length) then calculate  $N = p * q$ . ( $p$  and  $q$  are both private)
2. Calculate Euler Totient function  $\phi(N) = \phi(p) \phi(q) = (p - 1)(q - 1)$ .
3. Select an integer  $e$  (relative prime to  $\phi(N)$ ) and calculate  $d$  using Euclidean algorithm by the equation:  $e * d = 1 \pmod{\phi(N)}$ .

Then  $[e, N]$  are used as public key and  $[d, N]$  are kept as private key.

– **RSA Signing**

1. To sign a message  $m$ , use the private key  $[d, N]$  as in equation 2.

$$S \equiv m^d \pmod{N} \quad . . . . (2)$$

2. Then the signature  $s$  and message  $m$  are sent over to the receiver.

– **RSA Verify**

1. To verify the signature, the receiver uses the public key  $[e, N]$  as in equation 3.

$$v \equiv S^e \pmod{N} \quad . . . . (3)$$



2. If the calculated  $v =$  the received (m) then the sender is authentic and the message has integrity.

However, generally if RSA is used for signing a message, it is more efficient to sign a hash value of the message rather than the message itself.

### 2.4.3 ElGamal

This algorithm is related to D-H algorithm, where its backbone is the use of exponentiation in a finite field, and its security is based on the hardness of computing discrete logarithms. The advantage of ElGamal algorithm is that each time the same plaintext is encrypted, it gives different cipher text but with one disadvantage that the cipher text length is twice the length of the plain text. A private key is used to encrypt (sign) the message and a public key is used to decrypt (verify) the signature. The algorithm keys generation, signing and verification steps are listed below (ElGamal, T. 1985).

#### – ElGamal Keys Generation

1. Select a prime integer number  $p$ , (public).
2. Select a random integer number as  $g < p$  and it is a primitive root of  $p$ , (public).
3. Randomly select integer  $x$  as private key, such that  $1 \leq x \leq p - 1$ .
4. Calculate public key  $y \equiv g^x \bmod p$ , (public).

Therefore  $p$ ,  $g$  and  $y$  are public keys, but  $x$  is private key.

– **ElGamal Signing process**

The parameters  $p$ ,  $g$  and the calculated  $y$  are all made public.

To sign a message  $m$ , its hash value  $H(m)$  is calculated first. Then choose a random (or pseudorandom) integer  $k_i$  where  $k_i$  is also relative prime  $top - 1$ , which must be changed for each message, then sign the message by:

$$r \equiv g^k \text{mod} p \quad . . . . (4)$$

$$s \equiv (H(m) - x.r) k^{-1} \text{mod} (p - 1) \quad . . . . (5)$$

Where  $s$  must not be zero. Now  $r$  and  $s$  are the signature.

– **ElGamal Verification process**

$$1. \text{ Compute } V_1 \equiv r^s . y^r \text{mod} p. \quad . . . . (6)$$

$$\text{And } V_2 \equiv g^{H(m)} \text{mod} p. \quad . . . . (7)$$

2. If  $V_1 = V_2$  the signature is accepted, otherwise the message is not authentic.

## 2.4.4 Schnorr Digital Signature

This method is similar to DSA in some way like the use of Hash function and Key generator expect that there are no constraints to the size of  $p$  and  $q$ .  $p$ ,  $q$ ,  $g$  and  $y$  are made public and  $x$  is private as will be explained in the next subsection. The signature generation and verification are as follows (Menezes et.al, 1996):

– **Schnorr Signing process**

1. Select a random secret integer  $k$ ,  $1 \leq k \leq q - 1$ .

2. Compute  $r = g^k \text{mod} p$

3.  $e = h(m||r)$

4.  $s = ge + k \text{mod} q \quad . . . . (8)$

5. The signature of  $m$  is the pair  $(s,e)$ .

– **Schnorr Verification process**

1. Get the sender public keys (p,q, g,y).
2. Compute  $v = g^s y^{-e} \text{mod } p$ , and  $e' = h(m||v)$  . . . (9)
3. If  $e'=e$  then the signature is valid and will be accepted.

### 2.4.5 Digital Signature Algorithm (DSA)

DSA is a variant of ElGamal and Schnorr algorithms. It is the digital signature algorithm that is standardized by National Institute of Standards and Technology (NIST). It also employs public key, private key, and a hash function. The verification process is done using the same keys. The sender's public key y, which is associated with the private key x is mathematically used to generate the digital signature. In the verification process, a hash function should be used again to calculate the hash value of the message m. The block diagram of the DSA implementation scheme is shown in Figure (3). The algorithm parameters are described below:

– **DSA Key generation**

The following parameters are generated first with their criteria.

1. p: Large prime integer  $2^{L-1} < p < 2^L$ , L=the length of bits of (p).
2. q: Large prime integer divisor of (p – 1),  $2^{N-1} < q < 2^N$ , N=the length of (q).
3. g: subgroup generator of (q mod p),  $1 < g < p$ .
4. x: private integer,  $0 < x < q$  in range 1.. q – 1.
5. y: the signer's public key,  $y = g^x \text{mod } p$ .
6. k: secret random integer that is changed for each message (unique),  $0 \leq k \leq q$ .

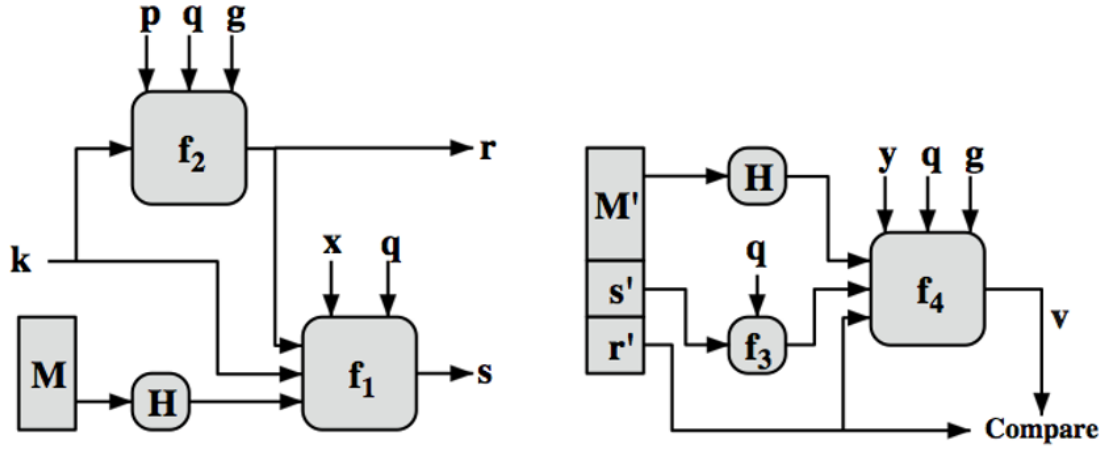


Figure (3) Block diagram of signing and verification processes of DSA

#### – DSA Signature Generation Process

The signature is generated for the hash value of the message  $m$   $H(m)$ , DSA use SHA1 as hash value, the calculated  $H(m)$  must be converted to an integer, then generate the signature of the message using the following equations (FIPS PUB 186-4, 2013).

$$r = f_1(k, g, p, q) = (g^k \bmod p) \bmod q \quad . \quad . \quad (10)$$

$$s = f_2(H(m), k, x, r, q) = (k^{-1}(H(m) + x \cdot r)) \bmod q \quad . \quad . \quad (11)$$

#### – DSA Verification process

The received signature is the pair  $r'$  and  $s'$ , so the verifier does the following

1. The verifier checks that  $0 < r' < q$  and  $0 < s' < q$ ; if either condition is breached then the signature shall be rejected because, it is invalid.
2. The verifier computes the following:

$$w = f_3(s', q) = (s')^{-1} \bmod q \quad . \quad . \quad . \quad . \quad . \quad . \quad (12)$$

$$v = f_4(y, q, g, H(m'), w, r')$$

$$= ((g^{(H(m') \cdot w) \bmod q} \cdot y^{(r' \cdot w \bmod q)}) \bmod p) \bmod q \quad . \quad . \quad . \quad . \quad (13)$$

3. If  $v = r'$  then the signature is verified.

### 2.4.6 Short Comparison

A brief comparison of the previous digital signature algorithms is listed in the following table.

Table (2.3) Comparison of Listed Algorithms in Chapter Two

Algorithm	Advantage	Disadvantage
D-H	<ul style="list-style-type: none"> <li>Challenging to solve discrete logarithms.</li> <li>The shared key never transmitted over the channel.</li> </ul>	<ul style="list-style-type: none"> <li>Involve in expensive exponential operation.</li> <li>Used to establish a secret key only and not to encrypt or sign a message</li> </ul>
RSA	<ul style="list-style-type: none"> <li>Key generator requires a lot of computation.</li> </ul>	<ul style="list-style-type: none"> <li>Slower than other symmetric cryptosystems.</li> </ul>
ElGamal	<ul style="list-style-type: none"> <li>Strong due to the change of random number (k), so every time the same plan text encrypted it will product new cipher text.</li> </ul>	<ul style="list-style-type: none"> <li>Need randomness and slow.</li> <li>Cipher text length is twice that of plan text.</li> </ul>
Schnorr	<ul style="list-style-type: none"> <li>Signature generation re require one exponentiation modulo and one multiplication modulo.</li> </ul>	<ul style="list-style-type: none"> <li>Dose not significantly enhances computational efficiency over the ElGamal scheme.</li> </ul>
DSA	<ul style="list-style-type: none"> <li>Signature does not depend on the message.</li> <li>Strong due to change of random key for each message</li> </ul>	<ul style="list-style-type: none"> <li>Probability of verification fail because of <math>S^{-1} \bmod q</math> if <math>S = 0</math>.</li> </ul>

### 2.4.7 The DSA Variants

Many scientists and mathematicians tried to improve implementation speed of the Digital Signature Algorithm. As a result of their studies many versions of DSA variants came up, some are faster than DSA in signature while others are faster in verification. Some of these variants have less complexity, while others are the same as

the original example of these variants are, Yen-Laih, GOST, McCurley, Ali, Naccache, et. al. Here are some of these variants.

### 2.4.7.1 GOST Digital Signature

A Russian version of digital signature algorithm is similar to the DSA also but it uses primes with specific length for the parameters p, q as follows (Schneier, B. 2000):

1. p: a prime number with length either between 509-512 or 1020-1024 bits.
2. q: a prime factor of p-1, between 254-256 bit length.
3. a (as g in DSA): a number less than p-1 such  $a^q \bmod p = 1$ .
4. x: a number less than q.
5.  $y = a^x \bmod p$ , as a public key.

– **GOST Signature process:**

1.  $\text{Calculator} = (a^k \bmod p) \bmod q$ , k is message secret random integer less than q.
2.  $s = (x \cdot r + k(H(m))) \bmod q$ .

– **GOST Verification process:**

1.  $v = H(m)^{q-2} \bmod q$ .
2.  $z_1 = (s \cdot v) \bmod q$ .
3.  $z_2 = ((q - r) * v) \bmod q$ .
4.  $u = ((a^{z_1} * y^{z_2}) \bmod p) \bmod q$ .

Accept the message signature if  $r = u$ .

### 2.4.7.2 Yen-Laih Digital Signature

A variant of DSA tries to make the signature faster by calculating the inverse of the fixed private key  $x$  in advance and use it for each signature, the Signature and verification processes are as follows:

#### – Yen-Laih Signature Generating Process

To sign a message the signer calculates the following:

1.  $r = (g^k \bmod p) \bmod q$
2.  $s = ((r \cdot k - h(m)) \cdot x^{-1}) \bmod q.$

#### – Yen-Laih Verification Process

To verify a signature in this algorithm the verifier has to calculate the following:

1.  $w = r^{-1} \bmod q.$
2.  $u1 = (w \cdot (h(m))) \bmod q.$
3.  $u2 = (w \cdot s) \bmod q.$
4.  $v = ((g^{u1} \cdot y^{u2}) \bmod p) \bmod q.$

if the calculated  $v$  is equal the revived  $r$  then the signature is accepted.

### 2.4.7.3 McCurley Digital Signature

In this algorithm an improvement of the DSA verification process is done by eliminating the inverse from the calculations on the verifier side in order to decrease the required time to verify the signature. The calculation of the signature and verification processes are as follows:

– **McCurley Signature Generating Process**

To generate signature in this algorithm the signer calculates the following:

1.  $r = (g^k \bmod p) \bmod q.$
2.  $s = (k \cdot (h(m) + x \cdot r^{-1}) \bmod q.$

– **McCurley Verification Process**

The verifier has to compute the following:

1.  $u1 = (h(m) \cdot s) \bmod q.$
2.  $u2 = (s \cdot r) \bmod q.$
3.  $v = ((g^{u1} \cdot y^{u2}) \bmod p) \bmod q.$

Accept the message if received  $r$  is equal to calculated  $v$ .

#### **2.4.7.4 Variant of DSA**

A modified version of standard DSA algorithm which is supported by the National Institute of Standards and Technology (NIST) will be investigated and compared with the original DSA in order to check its validity. This modified DSA version claims improvement in verification time relying on using less mathematical operations during the verification step while keeping the same security level of the process by using the same key strength and all calculations are based on the difficulty of solving Discrete Logarithmic problem in the multiplicative group of an appropriate Finite Field. It is achieved as shown below.



The signature and the validation functions are both modified with intention of have less computations. At the sender side, the calculation of signature  $s$  is modified and at the received side one equation of the DSA in verification calculation is omitted, and the verification equation is also modified to fulfill the signature validation criteria. These modifications have kept the same computation complexity level for the signature and verification of NIST-DSA but reduced the verification time (Ali, 2004). The original DSA algorithm was explained earlier in section 2.4.5. The function for calculating  $r$  (equation 10) is not changed but equation 10 becomes:

$$s = (k \cdot (x \cdot (H(m) + r))^{-1}) \bmod q \quad . . . . (14)$$

Calculating  $s$  in this way will make the verification calculations decreases due to the reduction in the exponentiations which makes generating the signature and verification steps faster i.e. more efficient than DSA. The receiver gets  $r$  and  $s$ , then validate the signature using the following functions.

1.  $u1 = (H(m) + r) \bmod q$
2.  $u2 = (s \cdot u1) \bmod q$
3.  $v = (y^{u2} \bmod p) \bmod q \quad . . . . . (15)$

Then if the value of  $v$  is equal to the received  $r$ , the signature is verified, but if they are not equal then the signature will be rejected. The block diagram is similar to that of Figure (3).

## 2.5 Message Digest

Message digest is a fixed length output value generated by hash functions for cretin message. Hash functions are one-way functions which mean that its output can't be inverted to recover the input, used to provide message integrity in order to ensure

that the receiver is receiving the same message sent by sender, and has not been modified during transmission, Message digest may be referred to as hash value or a checksum. There are many algorithms used to calculate the message digest, such as (Secure Hash Algorithm)SHA-1 (SHA128) that produces digest with 160-bit length, The input message consists of more than one block, for example an input with 512 bit is split into (80 x 32 bit) words, 32bit for each computation (mathematical and logical) round of the 80 rounds in the SHA-1, The 80 rounds are grouped into 20 rounds each one have different k and logical operations. Message digest or simply hash can be generated using other algorithms like SHA256, and (Message Digest algorithm) MD5 which are proposed by R.Rivest 1990-1991 and other algorithms. (Bajaj andGrewal,2015).

Hash functions must provide the following:

1. **Compression:** Hash functions generate (output) a fixed length of data whatever is the input length, and it is useful in generating signature for the message. When the message length is too long the signature will be huge and will take longer time than for smaller length of data representing this message. Hash function lengths range between 128 to 512 bits in general.
2. **Efficiency:** The function has to be capable to compute the hash value of any size of the input, keeping in mind that the time to generate the result according to the length of the input have to be acceptable.
3. **One Way:** As mentioned before, the Hash function must not be reversible computationally, one can get y from x, but the inverse getting x from y is not possible.

4. **Strength against weak collision:** For any given seed and its hash value like  $x$  and  $h(x)$ , it is computationally infeasible to find any  $w$  with  $w \neq x$  such  $h(w) = h(x)$ .
5. **Strong collision:** Hash functions will not generate two identical values for any two input data, i.e. each input data has unique output hash value.

SHA-1 algorithm will be implemented in the proposed work of this thesis prior to the calculation of the signature at the signers and verification at the verifier sides.

A simplified model example for the hash function is illustrated in Figure (4)

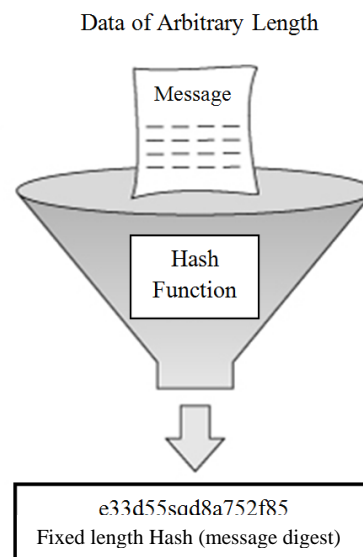


Figure (4) Hash Function

## **2.6 Blind Digital Signature**

It is a kind of digital signature that hides the content of the message from the signer, This type of digital signature is used nowadays in many fields like e-commerce, online payments and electronic voting system, In all of these systems the identity of the person who requests the signature is unknown, for example the online payments use blind signature so the bank do not know the payment for what reason, it just ensures that the money is legitimate i.e. when a person wants to buy from an online store with 50 JD for example, the person requests a signature on a check from the bank that this 50 JD is available, after getting the signature the person gives the check and the signature to the store, the store sends the check and the signature to the bank, then the bank will verify the signature using the bank public key, if the signature is verified then the bank will deduct 50 JD from the buyer's account and transfers the 50 JD to the store account, otherwise the transfer will stop. In such case the bank does not know what the buyer buys from the store or it was a donation or something else, and for the store, they don't know who the buyer is.

## **2.7 Group Digital Signature**

This type of digital signature consists of more than one person with unique signature and unknown by other members. In this type of digital signature a group of entities is given the ability to sign a document, and any member of this group can sign a document on behalf of the whole group, the document signature can be verified using group public key. In addition the identity of the signer is kept hidden except for the group manager who can trace the signer. Many recent projects require group signature like Trusted Computing effort and Vehicle Safety Communications project (VSC) (Shacham, 2005).

## 2.8 Group Blind Digital Signature

This type combines the properties of both the blind digital signature and group digital signature. It is useful to use when blind signature is used, most commonly it is used in transaction with electronic cash including multiple banks and online voting involving multiple voting servers (Ramzan, 1999).

## 2.9 Review of Related Literature

1. (Tan, Yi and Siew, 2003): studied the signature scheme based on composite discrete logarithm. they claims that their scheme is 13.5 times faster than RSA, 3 times faster than DSA and faster than some other scheme. It is also secure against existence forger using adaptive chosen message attack.
2. (Poulakis, 2009): present a digital signature algorithm that is a modified variant of DSA. It was based on a factorization problem. It implements two discrete logarithms and claims security strength at least equal to DSA and it withstands all known attacks. The represented work utilized the properties of RSA technique to introduce two modular exponentiations and a modular multiplication for the signature.
3. (Galego Hernandez Jr, Carvalho and Proenca Jr, 2014): They used Digital Signature to support Network management through traffic characterization to help network administrators. They use Digital Signature of network segment using flow analysis as a technique to describe standard network behavior, they also use genetic algorithm to optimize the process.
4. (Singh, Kaur and Kakkar, 2015): They proposed a scheme to detect any tampering and also support the image compression, they suggest that a digital signature transmitted along with the image itself, and the receiver will regenerate

the signature corresponding to the received image and if both signatures match then the received image is authentic.

5. (Tan, Tang, Chen, Yu and Li, 2016): proposed new approach to build a secure variant of rainbow (multivariate Digital Signature), and they claims that their variant can resist all the existing attacks according to their security analysis, and also that their scheme can save memory for both private key and public key than rainbow scheme.
6. (Andrade and Terada, 2016): present a new digital signature for Quartz digital signature (scheme based on Hidden Field Equations with vinegar - minus trapdoor), with special choice of parameters, which can resist algebraic attack to recover the private key and it was secure till present days, they claims efficiency gain in signature verification and using vector initialization for both signature and verification algorithms.
7. (Alpizer-chacon and Chacon-Rivas, 2016): used digital signature to solve the problem of verifying the authors the contents of learn objects in online education through different repositories, by uploading the learn object to repository, the author present his/her own digital certification, a special record created from this certification and signed to provide authenticity and trust overtime using digital signature infrastructure of Cost Rica.
8. (SadrHaghighi and Khorsandi, 2016): Used digital signature to detect pollution attack in intra session Network coding and proposed "Identity-based Digital Signature scheme", They claims that using this signature will make the intermediate nodes detect bogus packets and omit them before combining with other packets, In addition the sender can update its own keys without change of its

identity and it does not need certificate management. moreover the verification process is faster in their scheme compared with previous work.

9. (Dhagat and Joshi, 2016): They define a method for digital signature that uses another singer as proxy, this scheme provides the ability to the sender to delegate his signature to another signer, and claim that their scheme will provide protection to proxy signer private key, the scheme is ruled by certification holds identity of signer, delegate duration and imposes rules on the signing ability delegation by the original signer, so the proxy signer can sign instead of the original signer only within the validation period, the scheme uses protected nominative signature so that signer and proxy cannot deny each other.
10. (Manickam and Kesavaraja, 2016): They proposed a secure multi-server authentication system, utilizing advantages of digital signature in addition to elliptic curve, they claim that their system is secure against majority of possible active attacks, and provides high security along with low communication cost It establish simple structure and the task of an authentication system are analyzed with different EC metrics. Their proposed work claims to be more adoptable for smart cards and other less energy saving devices, the limitation of the system that the digital signature ensures the security of small size of messages. Their scheme depends on key size of elliptic curve, the message needs to be sent to digital signature authentication. The elliptic curve cryptography computation is based on scalar multiplication.

## **Chapter Three**

### **The Proposed Modified DSA**



## **Chapter Three**

### **The Proposed Modified DSA**

#### **3.1 Introduction**

One of the basic factors for digitally signing and verifying a message is the speed at which the execution of signing and verification process is accomplished. Obviously, the time complexity is related to the computational speed and mathematical relations involved. This chapter includes the investigation of the digital signature algorithm (NIST-DSA) for computation time measurements for signing and verification under various parameters variation first. Then a new modified version of DSA is suggested and investigated. It has improved the signature verification speed, as it reduces the time for signature validation. The same parameters of NIST DSA are employed but using different calculation arrangement, which have resulted in the obtained improvement. The mathematical proof for the proposed modified version is included too.

#### **3.2 Investigation of DSA algorithms methodology**

In this section, a test will be applied to both DSA and the modified variant (which will be referred to as M.DSA) that resulted into reduction in message signing and verification time complexity.

for these algorithms, signing and verification time is calculated using various values for random number  $k$  and a range of values for signer's private key  $x$ . The investigation methodology is as follows.

The first step in this work is the preparation of prime integers  $p$  and  $q$  (as described in section 2.4.5.1 of chapter 2), that both algorithms require to generate keys, first a prime of 160 bit is selected ( $q$ ), then a prime number ( $p$ ) is selected with length between 512 and 1024 bits such that  $q$  divides  $(p-1)$ . Then  $g$  is chosen, such that:

$g = h^{(p-1)/2} \bmod p$ , after explaining the rules of prime numbers, the steps of the parameters generation are explained as follows:

1. Generate prime numbers  $p, q$  and  $g$ .
2. Select Hashing algorithm to generate  $H(m)$ , in this work it is SHA1, where  $m$  is the message to be signed.
3. Select two random or pseudorandom integers to be the private key  $x$  and random number such that,  $0 < x \text{ and } k < q$ , then calculate the public key  $y$  as  $y = g^x \bmod p$ . The second integer is  $k$  which shall be changed for every signature.
4. Input the message ( $m$ ) and generate its hash value using algorithm in step 2.
5. Set number of required loops to 30 different private  $x$  and for each  $x$  also 30 different values for  $k$ .

6. Calculate the signature values  $r$  &  $s$  as:

$$r = (g^k \bmod p) \bmod q, \text{ and}$$

$$s = (k^{-1}h(m) + xr) \bmod q$$

7. Determine the time of signature generating ( $t_s$ ).
8. Calculate hash value of  $m$ , and calculate  $v$ , and determine the time of calculating  $v(t_v)$ .
9. Store or print value of  $(x, k, t_s, t_v)$ , change  $k$ .
10. Repeat a new loop with new value of  $x$  and  $k$  until the required number stated in step 4 is achieved as explained in the flow chart for Figure(5).

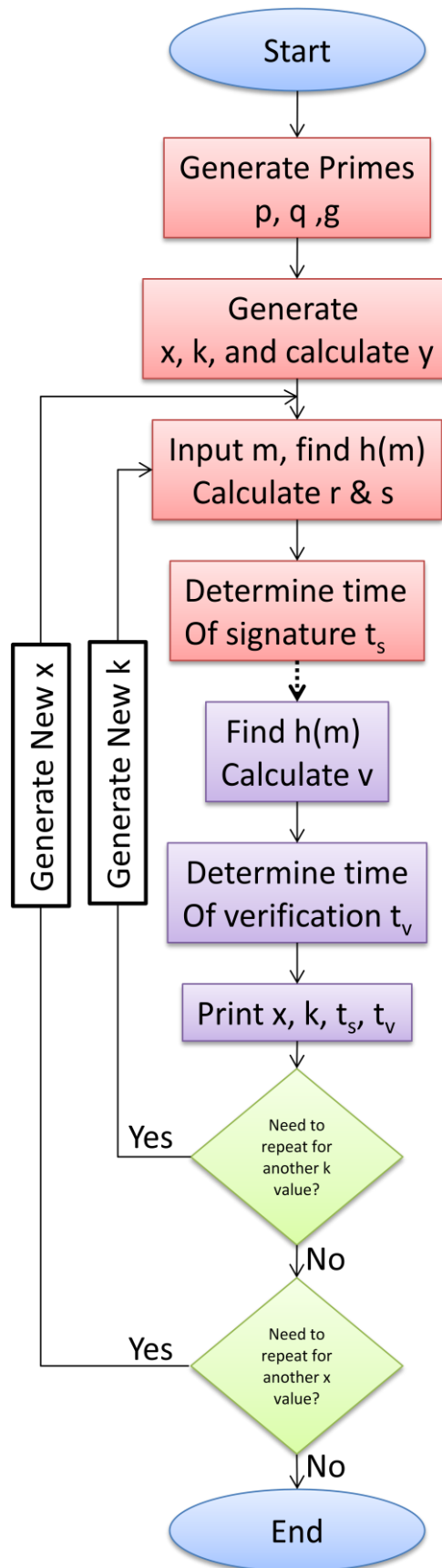


Figure (5) Flowchart for measuring the time complexity for the algorithm.

### 3.3 The proposed Modified DSA (M.DSA)

A modification to the digital signature algorithm is proposed here with the aim to increase the strength and speed of the digital signature scheme, by increasing the key strength and the process of using them. The modification proposed in this work also adds some complexity into the key generation.

This work is intended to modify both the DSA signing and verification protocols by introducing new modification of digital signature. It is intended to be faster in both sign and verification calculations.

At the signing side, and after the calculation of the message hash,  $H(m)$  and generating random number  $k$  as in the NIST-DSA algorithm, the signature parameter  $r$  and  $s$  are calculated as follows:

- Choose a random number  $k$  with  $k: 1 \leq k \leq q - 1$ .
- Compute  $r1 = (g^k \bmod p) \bmod q \dots (16)$
- Compute  $h = H(M)$ .
- Compute  $s = r + k(x.h)^{-1} \bmod q \dots (17)$
- Alice signature for the message  $M$  is  $(r, s)$ .

Notice the modification is only to the calculation of  $s$  while  $r$  is the same as before.

Now, signature  $r$  and  $s$ , together with the message  $m$  are sent to the recipient at the receiver side, verification of the signature takes place in the following steps, which are completely different from that of NIST-DSA as follows:

- Verify that  $r, s$  are integers in the interval  $[1, q - 1]$ .
- Compute  $h = H(M)$ .
- Compute  $u1 = s * h \bmod q$ . ...(18)
- Compute  $u2 = r * h \bmod q$ . ...(19)
- Compute  $v = (y^{u1-u2} \bmod p) \bmod q$ . ...(20)
- Accept the signature if and only if  $v = r$ .

Therefore, if the signature ( $r$  and  $s$ ) of the message  $m$  was indeed generated by signer,  $v = r$ , otherwise either the sender is not authentic or the message itself has been changed in transit.

The research work will be conducted according to the following procedure

1. Prove that  $v = r$  for valid signature mathematically.
2. Implemented the suggested modification and test it.
3. Repeat the calculations for variety of different values for key  $x$  and random numbers  $k$  &  $d$ .
4. Compare the obtained results with those obtained for the standard DSA and the variants.

For the proposed digital signature algorithm, and all others under considerations, three phases are followed in order to be implemented and tested, as shown in Figure(5)

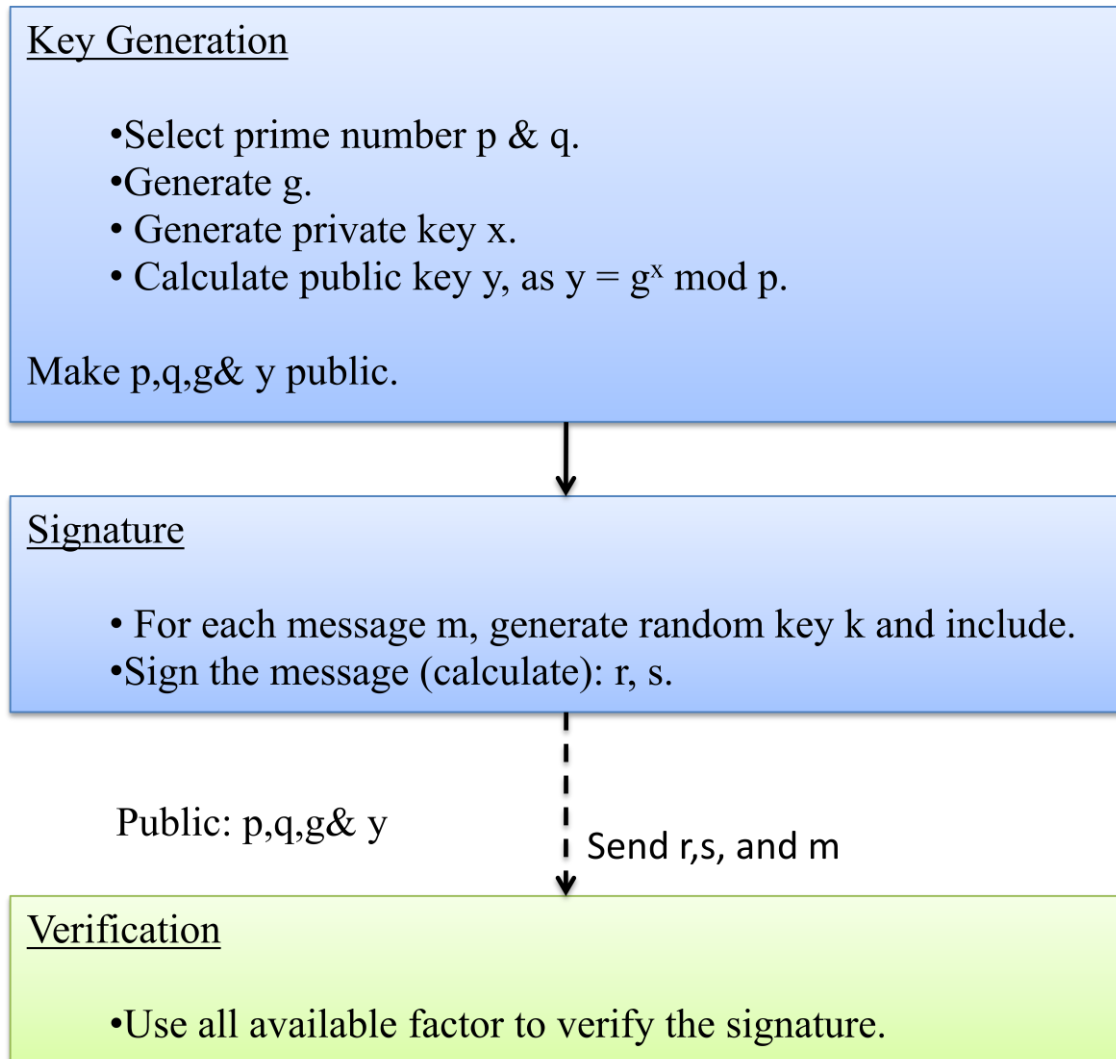


Figure (6) the steps followed in the modified algorithm

### 3.4 The modified M.DSA proof

In order to be satisfied with the correctness of the Modified DSA (M.DSA) mathematical modification introduced in this thesis. A proof for the M.DSA is in the lemma below:

Lemma1: For the signature verification to be true, the value of r must equal v.

Therefore, from the exponent of r and v the following must be true:

$$k = x \cdot u_1 - x \cdot u_2$$

Proof

The verifier ( message receiver ) calculates,  $H(m)$ ,  $u_1$ , and  $u_2$  as follows:

$$u_1 = s H(m) \bmod q,$$

$$u_2 = r H(m) \bmod q.$$

$$\text{Now } v = (y^{u_1 - u_2} \bmod p) \bmod q$$

Substitution  $y = (g^x \bmod p) \bmod q$  in  $v$  gives:

$$v = g^{x(u_1 - u_2)} \bmod p \bmod q$$

$$v = ((g^{xu_1 - xu_2}) \bmod p) \bmod q$$

Comparing this equation with  $r$

$$r = (g^k \bmod p) \bmod q$$

Therefore for the last two equations to be true, the following must be true

$$g^k = ((g^{xu_1 - xu_2}) \bmod p) \bmod q$$

Therefore  $k$  must equal the exponent of  $g$  i.e.  $k = x(u_1 - u_2)$

Substituting for  $u_1$  and  $u_2$

$$k = (x(s.H(m) - r.H(m)) \bmod p) \bmod q$$

$$= H(m)x(s - r) \bmod p \bmod q$$

Substitution of  $s$  gives:

$$k = H(m)x(r + k.(x.H(m))^{-1} - r) \bmod p \bmod q$$

$$= x.H(m).k.(x.H(m))^{-1} \bmod p \bmod q$$

$$= (k \bmod p) \bmod q = k$$

Thus,

$$g^k = y^{u_1 - u_2}$$

so  $v = r$  as required. (q.e.d)



The proof of NIST-DSA will be listed with the description given in the next section.

### 3.5 Digital Signature Algorithm

The U.S. National Institute of Standards and Technology (NIST) proposed a digital signature algorithm (DSA), in August of 1991. The DSA has become a U.S. Federal Information Processing Standard (FIPS 186). It is the first digital signature technique presented by any government. The digital signature mechanism requires a hash function. The digital signature explicitly requires use of the Secure Hash Algorithm.

Digital Signature Standard (DSS) is the analogue to handwritten signatures. A digital signature is a number depends on the secret information only known by the signer and on the contents of the message being signed. Signatures must be verifiable without access to the signer's secret information. This asserts that an adversary who is able to obtain signatures for any messages of his choice cannot forge his signature on a single other message.

Suppose Alice wants to send a digitally signed message to Bob. They first choose a prime  $q$  and the generator  $g$  is an element of prime order  $q$ . The parameters  $p$  and  $q$  are primes, the  $q$  must divide  $p - 1$ . Alice's key pair is  $(x, y)$ , where  $x$  is his secret and  $y$  is his public key. To sign a message  $M$  Alice does the following:

- Choose a random number  $k$  with  $k: 1 \leq k \leq q - 1$ .
- Compute  $r1 = g^k \bmod p$ .
- Compute  $r = r1 \bmod q$ .
- Compute  $k^{-1} \bmod q$ .
- Compute  $h = H(M)$ .

- Compute  $s = k^{-1}(h + x.r) \bmod q$ .
- Alice signature for the message M is  $(r, s)$ .

To verify Alice's signature  $(r, s)$  on the message M, Bob obtains an authentic copy of Alice's parameters and public key. Bob should validate the obtained parameters. Bob then does the following:

- Verify that  $r, s$  are integers in the interval  $[1, q - 1]$ .
- Compute  $h = H(M)$ .
- Compute  $w = s^{-1} \bmod q$ .
- Compute  $u_1 = h.w \bmod q$ ,
- Compute  $u_2 = r.w \bmod q$ .
- Compute  $v = (g^{u_1} \cdot y^{u_2} \bmod p) \bmod q$ .
- Accept the signature if and only if  $v = r$ .

If the signature  $(r, s)$  on the message M was indeed generated by Alice, then

$s = k^{-1}(h + x.r) \bmod q$ . With this information:

$$\begin{aligned} k &= s^{-1}(h + x.r) \bmod q \\ &= s^{-1}h + s^{-1}x.r = w.h + w.x \\ &= u_1 + u_2.x \bmod q. \end{aligned}$$

Thus

$$g^{u_1} \cdot y^{u_2} = g^{(u_1 + u_2 x)} = g^k$$

so  $v = r$  As required.

The following algorithm describes the digital signature algorithm.

### 1. Initialization

- Alice and Bob publicly choose primes  $q$  and  $p$ , where  $q$  divide  $p-1$ .
- They publicly choose a number  $g$ , where  $1 < g < q - 1$ .

### 2. Key generation

- Choose a secret key  $x$ , where  $1 < x < q - 1$ .
- She then computes  $y = g^x \bmod p$ .
- Make  $y$  public and keep  $x$  secret.

### 3. Signature generation

Alice sends the digitally signed message to Bob as follows

- Select random integer  $k$ , where  $0 < k < q$ .
- Compute  $r = (g^k \bmod p) \bmod q$ .
- Compute  $h = H(M)$ .
- Compute  $s = k^{-1} (h + x.r) \bmod q$ .
- The signature for  $M$  is  $(r, s)$ .

### 4. Signature verification

Bob verifies Alice's signature  $(r, s)$  on message  $M$  as follows

- Compute  $h = H(M)$ .
- Compute  $w = s^{-1} \bmod q$ .
- Compute  $u_1 = h.w \bmod q$ .
- Compute  $u_2 = r.w \bmod q$ .
- Compute  $v = (g^{u_1} . y^{u_2} \bmod p) \bmod q$ . Accept the signature if  
and only if  $v = r$ .

Example of application of the DSA algorithm

### 1. Initialization

- $q = 937$ .
- $p = 26237$ .
- $g = 9853$ .

### 2. Key generation

- $x = 747$ .
- $y = (g^x \bmod p)$   

$$= (9853^{747} \bmod 26237)$$

$$= 3541$$

Make  $y$  public and keep  $x$  secret.

### 3. Signature generation

Alice sends the digitally signed message to Bob as follows

- $k = 511$ .
- $r = (g^k \bmod p) \bmod q$   

$$= (9853^{511} \bmod 26237) \bmod 937$$

$$= 601.$$
- Let  $h = 1000$ .
- $s = k^{-1} (h + x r) \bmod q$   

$$= 926 (1000 + 747 * 601) \bmod 937 = 754$$

The signature for  $M$  is  $(r, s) = (601, 754)$ .

#### 4. Signature verification

Bob verifies Alice's signature  $(r, s)$  on message  $M$  as follows

- Let  $h = 1000$ .
- $w = s^{-1} \bmod q$   
 $= 754^{-1} \bmod 937 = 128$ .
- $u_1 = h \cdot w \bmod q$ .  
 $= 1000 * 128 \bmod 937$   
 $= 568$ .
- $u_2 = r \cdot w \bmod q$   
 $= 601 * 128 \bmod 937$   
 $= 94$
- $v = (g^{u_1} * y^{u_2} \bmod p) \bmod q$   
 $= (9853^{568} * 3541^{94} \bmod 26237) \bmod 937$   
 $= (24026) \bmod 1511$   
 $= 601$

Accept, where  $v = r = 601$ .

Example of application of the M.DSA algorithm

### 1. Initialization

- o  $q = 937$ .
- o  $p = 26237$ .
- o  $g = 9853$ .

### 2. Key generation

- o  $x = 747$ .
- o  $y = (g^x \bmod p)$   
 $= 9853^{747} \bmod 26237$   
 $= 3541$ . Make  $y$  public and keep  $x$  secret.

### 3. Signature generation

- o  $k = 511$ .
- o  $r = (g^k \bmod p) \bmod q$ .  
 $= (9853^{511} \bmod 26237) \bmod 937$   
 $= 601$ .
- o Let  $h = 1000$ .
- o  $s = r + k * (xh)^{-1} \bmod q$ .  
 $= (601 + 511 * 675) \bmod 937$   
 $= 710$ .
- o The signature for  $M$  is  $(r, s) = (601, 710)$ .

#### 4. Signature Verification

Bob verifies Alice's signature (r, s) on message M as follows

Let  $h = 1000$ .

$$\begin{aligned} u_1 &= s.h \bmod q \\ &= (710 * 1000) \bmod 937 \\ &= 691 \end{aligned}$$

$$\begin{aligned} u_2 &= r.h \bmod q \\ &= (601 * 1000) \bmod 937 \\ &= 383 \end{aligned}$$

$$\begin{aligned} v &= (y^{u_1 - u_2} \bmod p) \bmod q \\ &= (3541^{691 - 383} \bmod 26237) \bmod 937 \\ &= 601 \end{aligned}$$

Accept, where  $v = 601$ .

### 3.6 Security Concerns of M.DSA

M.DSA is a digital signature algorithm based on DSA, this algorithm calculates the signature for the hash value of the message (r, s) and verifies the signature using public key of the sender as proved in (3.4) to enable the signer sending it with the message to provide authentication and to guarantee the source of the message to the receiver. Since the digital signature is equivalent to the hand written signature, the digital signature do not encrypt the signed message, but use mechanism to authenticate the sender and to ensure that the message was not altered during transmission, that means the digital signature provides integrity for the message. Using the private key and secret random integer that is changed for each message then verification is done to this

signature by only the sender's public key, and so makes the sender cannot deny his signature on the message, which provide non-repudiation.

### 3.7Key Strength

The strength of digital signature algorithms is depending on the length of the prime integers, the length of the private key and the length of the randomly generated secret integer. The length in bits for standard DSA for p is between 1024 to 3072 bits, and for q is 160 to 256 bits. Nowadays the speed of the computers allows using larger values of the primes p and q which are recommended to provide higher security against the attackers.

For example if private key x length and the random number integer k length are taken to be 64 decimal digit.

∴ The maximum number of possible different private key x is  $= 10^{64}$  and it is the same for k.

∴ The total number of possible keys  $= 10^{64} * 10^{64} = 10^{128}$ .

Using brute force attack, one may get the correct key in the first trial or possibly in the last trial, therefore, the average number of trials  $= \frac{1+10^{128}}{2} = 5 * 10^{127}$ .

Now if one assumes that each possible key trial takes  $10^{-10}$  seconds, then the average time required to find the correct key  $= 5 * 10^{127} * 10^{-10}$

$$= 5 * 10^{117} \text{ Seconds}$$

$$= 5787 * 10^{100} \text{ days}$$

$$= 15.85 * 10^{100} \text{ years}$$

Therefore, this digital signature scheme is considered really secure and authentic.



## **Chapter Four**

### **Implementation and Results**

## **Chapter Four**

### **Implementation and Results**

#### **4.1 Introduction**

This chapter includes the implementation of DSA and the proposed modified DSA algorithm, in addition to another four algorithm variants. The variants that will be investigated and compared with are GOST, Yen-Laih, McCurley and VAR-DNA. Microsoft Visual Studio Integrated Development Environment using C# language is used for the coding and testing. The chapter also will include results, comparison, and calculation for BigO of the algorithms.

#### **4.2 Digital Signature Implementation**

This section includes the digital signature implementation for both the standard DSA algorithm and the modified M.DSA algorithm in order to be able to investigate and compare the modified one with the standard DSA for various values of the private key of the signer and for various values of the random integer number  $k$  used for each message. This study is conducted first for values of the primes  $p$ ,  $q$  and integer  $h$  of about 100 decimal digit length the value of  $g$  is calculated and all these values are fixed as listed in table 4.1.

**Table (4.1) Parameters of Algorithms with Primes Length of 100 Decimal Digits.**

<b>p</b>	46513743332777904273926845393782128569537431460690416058313774312284 125238113922135346528273281049567
<b>q</b>	17889901281837655489971863612993126372899012100265544637812990120109 27893773612389821020318203117291
<b>h</b>	18770538827352476665248260551708880946519589543298512941727986691093 05505220810669990674063180575605
<b>g</b>	35380062791882015598414765785538569072394399144506415046676880717767 615983413224376710926111694831716

The signing time and the signature verification time are calculated for different values of the private key  $x$ , (namely  $x = 20, 40, 60, 80$ , and  $100$  decimal digit).

At the same time for each value of  $x$ , the average time is calculated for 10 different values of the random integer number  $k$ . This test is done to see the effect of key length on the time measurement. The results are tabulated in tables 4.2 and 4.3. The length of the  $k$  for each case is taken of the same length of  $x$ . The full tables are listed in Appendix A.

**Table (4.2) Signing and Verification Time Calculation for DSA, with  $p$ ,  $q$  and  $g$  of Length of 100 Digit, 10 Values of  $k$  For Each  $x$  Value.**

No.	X	Average Signature time in ( $\mu$ sec)	Average Verification time ( $\mu$ sec)
20	69262581357724378245	8388.7	56835
40	1458119622525378455431743088583285857803	14803.8	67352.3
60	3958215979062075272660808664834831686396865 29409348039394197	19606.3	56644.5
80	2449307626745069635533521987238783026039795 0085937388566084156030964862538005864	22509.4	50548.6
100	1714426734224640718858096300164316123816109 1579176853243017139454329676796364885082937 82726079154249	22438.1	42678.8

**Table (4.3) Signing and Verification Time Calculation for M.DSA, with p, q and g of Length of 100 Digit, 10 Values of k for Each x Value.**

No.	X	Average Signature time (μsec)	Average Verification time (μsec)
20	84252756894206064432	8266.3	17475.6
40	7400016487469127858591431200972451040017	16027.8	22117.8
60	9619142707144576197123005887824323859070471 69230011597205532	12726.7	16126.2
80	7417146235993325418957223410006593183690238 3993482619108885167982374298374293812	12181.1	15869.2
100	4801212400714631772596191427071445761971230 0588782432385907047169230011597205509282366 2890083108500	18051.6	16169.3

From tables 4.2 and 4.3, the differences in computation time for DSA and M.DSA are obvious. It's shown that the proposed M.DSA is faster. However, from these tables it was clear that the calculated average time using different keys and different secret message random integer numbers do not follow a certain pattern, rather they may go up or down due to the computer internal time complexity. Therefore it is decided that the average of time calculation would be more comprehensive if it is taken for large number of calculating rounds. Hence this will be considered in the next section and will be adopted for the M.DSA, DSA and all its variants under consideration.

### 4.3 Digital Signature Comparison

The average signing, verification, and total computation time is calculated for various digital signature algorithms under consideration using large number of rounds (namely 900 different keys), the algorithms that are considered for the proposed algorithm in this thesis (i.e. M.DSA) to be compared with are: DSA (NIST), Yen-Laih (Yen, S. M., & Laih, 1995), GOST (Bruce, S. ,1996), McCurley (Schnier,1996) and Variant of DSA (Ali, 2003).

The results summary of the calculations are listed in table (4.4). While the detailed calculation results for each algorithm are listed in section 4.5

Table (4.4) An Average of ((900 Signature and 900 Verify) \* 15 Round) Time in ms  
Compare for Several Algorithms with M.DSA.

No.	Algorithm	Avg. Signature Time ( $\mu$ sec)	Avg. Verify Time ( $\mu$ sec)
1	GOST	13131.03689	41062.0943
2	McCurley	15242.13333	27255.33333
3	Yen-Laih	15220.92385	29286.98578
4	Variant of DSA	15181.52926	13493.13452
5	DSA	15165.72104	29092.58304
6	M.DSA	15443.97437	13644.07741

A histogram chart will possibly show these results more clearly, as shown in Figure(7).

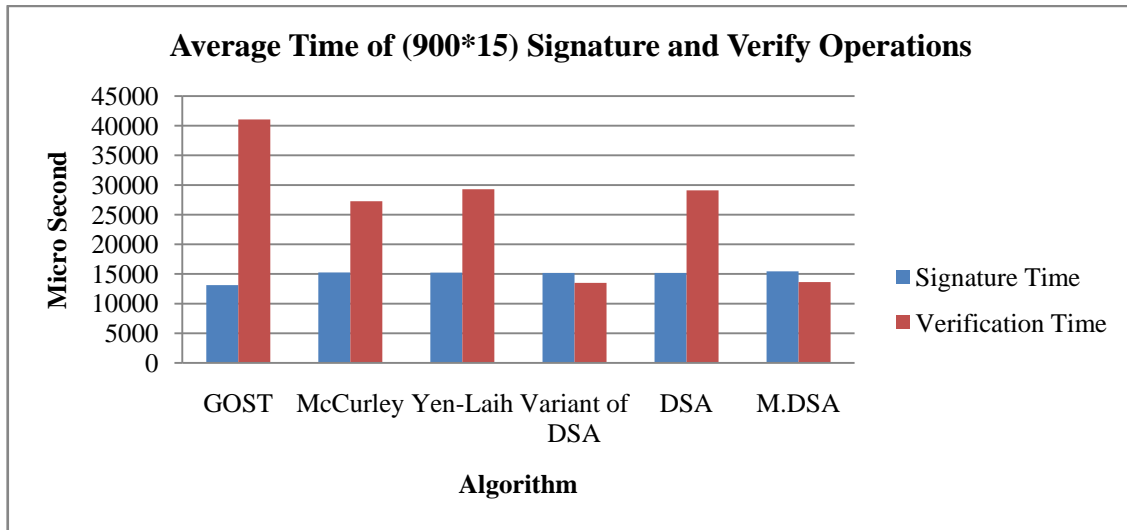


Figure (7) Comparison of total time of Variants of DSA with M.DSA

## 4.4 Speed Gain

The improvement in the signing and/or the verification time computation of M.DSA, to any other algorithms can be calculated by dividing that algorithm average time by the M.DSA average time. In the following values of speed gains of M.DSA over other algorithms are shown.

### 4.4.1 Signature Speed Gained

Like other attempts to modify DSA for better speed, success in speed gain is one goal (However one cannot gain for all goals such as make both sign and verify faster). The speed gain of M.DSA over other DSA algorithms is calculated by the following equation.

M.DSA and DSA

$$\text{speed gained} = \frac{\text{DSASignaturetime}}{\text{M.DSASignaturetime}} = \frac{15165.7}{15443.97} = 0.98$$

Likewise the signing speed gain is calculated for all other variants and the results are summarized in table (4.5)

**Table(4.5) Signature Speed Gained For M.DSA Over Other Algorithms.**

No.	Algorithm	Avg. Signature time (μsec)	M.DSA Avg. Signature time (μsec)	M.SDA speed over other algorithm
1	DSA	15165.72	15443.97	0.98
2	GOST	15181.53	15443.97	0.85
3	McCurley	13131.04	15443.97	0.98
4	Yen-Laih	15242.13	15443.97	0.98
5	Variant of DSA	15220.92	15443.97	0.98

#### 4.4.2 Verify Speed Gained

The M.DSA is providing more efficiency in verify time than DSA and some other algorithms. The calculations of the speed gain for verification for M.DSA over DSA is calculated by the following equation.

M.DSA and DSA:

$$\text{speed gained} = \frac{DSA_{\text{verifytime}}}{M.DSA_{\text{verifytime}}} = \frac{29029.6}{13644.08} = 2.13$$

**Table (4.6) Verify Speed Gained of M.DSA Over Other Algorithms.**

No.	Algorithm	Avg. Verify time (μsec)	M.DSA Avg. Verify time (μsec)	M.SDA speed over other algorithm
1	DSA	29092.6	13644.08	2.13
2	GOST	41062.09	13644.08	3.00
3	McCurley	27255.33	13644.08	1.99
4	Yen-Laih	29286.99	13644.08	2.14
5	Variant of DSA	13493.13	13644.08	0.98

## 4.5 DSA Variants Implementation

The DSA variants, namely GOST, Yen-Laih, McCurley and VAR-DSA algorithms coded and investigated for computation of message signing and verification.

They are tested on the same computation environment as NIST-DSA and M.DSA using the same public keys  $p$ ,  $q$ ,  $g$ , and  $y$  and about same length of private key  $x$  and random number  $k$  the detailed results are listed in tables (4.7), (4.8), (4.9) and (4.10).

**Table (4.7) VAR-DSA Algorithm.**

Length	x	Average Signature time ( $\mu$ sec)	Average Verification time ( $\mu$ sec)
20	60532331137761818651	10401.4	24192.8
40	970019126641409614021502424467675739054 1	8816.4	14807.4
60	668976577264078195731954814062470598315 618634149928347598234	12769.8	17095.3
80	485605323311377618186515101251338682837 657236830174340681351732998293475928342 34	21641.5	21624.2
100	334899012818376554899798210203182031172 915690723943991443221366677037417753881 2065898098092384234012	15514.2	13367.9



**Table (4.8) GOST Algorithm**

Length	x	Average Signature time (µsec)	Average Verification time (µsec)
20	33133806449883796951	3778.9	46674.4
40	559831561863414925493580832689190735840 7	8392.2	54206.7
60	513841687536145274291344678375347801670 732413788953979861997	9042.7	44468.8
80	885049147105152815618265098512411497429 653455345234534523198009464076086408098 98	12027.2	44449.8
100	770370792994367385408314615021230513619 420019350523089076239567506747872673138 5973952712663570744432	15212.3	43379.5

**Table (4.9) McCurley Algorithm**

Length	x	Average Signature time (μsec)	Average Verification time (μsec)
20	50212305136194200193	6826.5	31580
40	129932989544977626785365064899812742981 9	11285.6	30308.9
60	968080166029800979077409169497294786274 316857264078195731954	11409.7	29131.5
80	738540831461502123051361942001935052308 907623956750674787267313859739527126635 70	24815.6	48923
100	139469475024607592717989185681189259945 653400562860659723016352012326678983597 3098708003270130422769	18776.2	30975.1

**Table (4.10) Yen-Laih Algorithm**

Length	x	Average Signature time (μsec)	Average Verification time (μsec)
20	57470774235196347829	4875.3	35423.4
40	208792708840725619938497756294163490493 0	7915.7	35162.4
60	301172832362472388676302499954032882857 666917488448579338212	13377.1	41405.6
80	956843126650491844614363011728323624723 886763024999540328828576669174884485793 38	14918.3	35454.9
100	116890327800273283813485766488299107816 052805672364866178837642131930001501148 5638707775687531670877	18507.1	33130.8

## 4.6 Algorithms Signature and Verify complexity

The time complexity is one of the most important measures for cryptographic algorithm development. The most popular usage measure is the Big O notation. This classifies the cryptographic algorithm into the main classes of algorithm time complexity. In this section the measure of each Digital Signature Algorithm will be presented

DSA signature of the message hash consists of two values  $r$  and  $s$ . A complexity computation will be performed for both of them for DSA, M.DSA, Variant of DSA, GOST, McCurley, and Yen-Laih algorithms. Then an overall complexity of the signature will be performed for all of algorithms signature complexity.

### 4.6.1 DSA signature complexity:

$r$  value complexity:

$$r = (g^k \bmod p) \bmod q$$

$$O(r) = O((g^k \bmod p) \bmod q).$$

$$O(r) = O(\log n) \text{ according to fast power algorithm.}$$

$s$  value complexity:

$$s = (k^{-1} \cdot (H(m) + x \cdot r)) \bmod q$$

$$O(s) = O(k^{-1} \cdot (H(m) + x \cdot r) \bmod q).$$

$$O(s) = O(\log n) \text{ according to Extended Euclid's Algorithm.}$$

The power operation is the most time cost in the signature generation to compute  $y$ , where, the complexity of  $r$  computing is  $O(\log n)$  according to fast power algorithm. The inverse operation is the most cost operation in the signature generation where, the complexity of  $s$  computing is  $O(\log n)$  according to Extended Euclid's Algorithm used to compute the inverse operation. Individually, the signature has one power, one inverse operation, one addition operation, two multiplication, and three Mod operations. The addition, multiplication and Mod operations are considered with little or no cost compared to power and inverse operation.

Therefore,

$$O(DSA\ Signature) = O(2 \log n).$$

#### 4.6.2 DSA Verify complexity

DSA verify calculations contain four calculation steps as follows:

$$w = s^{-1} \bmod q$$

$$u1 = (H(m).w) \bmod q$$

$$u2 = (r.w) \bmod q$$

$$v = ((g^{u1}.y^{u2}) \bmod p) \bmod q$$

$O(w) = O(s^{-1} \bmod q), = O(\log n)$  according to Extended Euclid's Algorithm.

$O(u1) = O((h(m).w) \bmod q), = O(1)$  one multiplication operation of  $h(m)$  with  $w$ .

$O(u2) = O((r.w) \bmod q), = O(1)$  one multiplication operation of r and w.

$$O(v) = O((g^{u1} . y^{u2} \bmod p) \bmod q)$$

$$O((g^{u1} * y^{u2} \bmod p) \bmod q) = O(O(g^{u1}) . O(y^{u2}))$$

$$= O(\log n) + O(\log n) + O(1), \text{ according to fast power Algorithm.}$$

The  $O(u1)$  and  $O(u2)$  and any operation of  $O(1)$  such as Mod will be neglected.

Therefore,

$$O((g^{u1} * y^{u2} \bmod p) \bmod q) + O(w) + O(u1) + O(u2) = O(3 \log n).$$

#### 4.6.3M.DSA signature complexity

$$r = (g^k \bmod p) \bmod q$$

$$O(r) = O((g^k \bmod p) \bmod q).$$

$$O(r) = O(\log n), \text{ according to fast power algorithm.}$$

$$s = (r + k(x.h(m))^{-1}) \bmod q.$$

$$O(s) = O((r + k(x.h(m))^{-1}) \bmod q)$$

$$O(s) = O(\log n), \text{ according to Extended Euclid's Algorithm}$$

The M.DSA signature operation has one power, one inverse operation, one addition operation, two multiplication, and three Mod operations. The addition,

multiplication and Mod operations considered with no cost according to power and inverse operation.

Therefore,

$$O(\text{Signature}) = O(2 \log n)$$

#### 4.6.4M.DSA Verification complexity

M.DSA has three steps for verify:

$$u1 = s . h(m) \bmod q.$$

$$u2 = r . h(m) \bmod q.$$

$$v = (y^{u1-u2} \bmod p) \bmod q.$$

$$O(v) = O((y^{u1-u2} \bmod p) \bmod q) + O(u1) + O(u2).$$

$$O(u1) = O(1), \text{ where it is one multiplication of } s \text{ with } h$$

$$O(u2) = O(1), \text{ where it is one multiplication of } r \text{ with } h$$

$$O((y^{u1-u2} \bmod p) \bmod q) = O(\log n), \text{ according to fast power}$$

Algorithm

The  $O(u1)$  and  $O(u2)$  and any operation of  $O(1)$  such as Mod will be ignored.

Therefore,

$$O(v) = O((y^{u1-u2} \bmod p) \bmod q) + O(u1) + O(u2)$$

$$O(v) = O(\log n)$$

#### 4.6.5 GOST Signature complexity

$$r = (a^k \bmod p) \bmod q$$

$$O(r) = O((a^k \bmod p) \bmod q).$$

$$O(r) = O(\log n) \text{ according to fast power algorithm.}$$

$$s = (x.r + k.(h(m))) \bmod q$$

$$O(s) = O((x.r + k.(h(m))) \bmod q).$$

$$O(s) = 1.$$

Therefore,

$$O(\text{signature}) = O(\log n)$$

#### 4.6.6 GOST Verify complexity

GOST verification has four steps as follows:

$$v = h(m)^{q-2} \bmod q.$$

$$z1 = (s.v) \bmod q.$$

$$z2 = ((q - r).v) \bmod q.$$

$$u = ((a^{z1} . y^{z2}) \bmod p) \bmod q.$$

$$O(v) = O(h(m)^{q-2} \bmod q).$$

$$O(v) = (\log n)$$

$$O(z1) = O((s.v) \bmod q)$$

$$O(z1) = 1.$$

$$O(z2) = O(((q - r).v) \bmod q)$$

$$O(z^2) = 1.$$

$$O(u) = O((a^{z^1} \cdot y^{z^2}) \bmod p) \bmod q$$

$$O(u) = (\log n) + (\log n) = O(u) = O(2 \log n)$$

Therefore,

$$O(\text{Verify}) = O(u) + O(v) = O(\log n) + O(2 \log n)$$

$$O(\text{verify}) = O(3 \log n)$$

#### 4.6.7Yen-Laih Signature Complexity

$$r = (g^k \bmod p) \bmod q$$

$$O(r) = O((g^k \bmod p) \bmod q).$$

$$O(r) = O(\log n), \text{ According to fast power algorithm.}$$

$$s = ((r \cdot k - H(m)) \cdot x^{-1}) \bmod q$$

$$O(s) = O(((r \cdot k - H(m)) \cdot x^{-1}) \bmod q).$$

$$O(s) = O(\log n)$$

Therefore,

$$O(\text{signature}) = O(\log n) + O(\log n)$$

$$O(\text{signature}) = (2 \log n)$$



#### 4.6.8 Yen-Laih Verification Complexity

Yen-Laih algorithm has four steps to verify signature, as follows:

$$w = r^{-1} \bmod q,$$

$$u1 = (w \cdot (H(m))) \bmod q,$$

$$u2 = (w \cdot s) \bmod q$$

$$v = ((g^{u1} \cdot y^{u2}) \bmod p) \bmod q$$

$$O(w) = O(r^{-1} \bmod q)$$

$$O(w) = O(\log n), \text{ according to Extended Euclid's Algorithm}$$

$$O(u1) = O((w \cdot (H(m))) \bmod q)$$

$$O(u1) = 1, \text{ One Multiplication of } w \text{ with } H(m)$$

$$O(u2) = O((w \cdot s) \bmod q)$$

$$O(u2) = 1, \text{ One Multiplication of } w \text{ with } s.$$

$$O(v) = O(((g^{u1} \cdot y^{u2}) \bmod p) \bmod q)$$

$$O((g^{u1}) + O(y^{u2})) = O(\log n) + O(\log n)$$

$$O(v) = O(2 \log n).$$

Therefore,

$$O(\text{verify}) = O(w) + O(v) = O(\log n) + O(2 \log n)$$

$$O(\text{verify}) = O(3 \log n)$$

#### 4.6.9 McCurley Signature Complexity

$$r = (g^k \bmod p) \bmod q.$$

$$O(r) = O((g^k \bmod p) \bmod q)$$

$$O(r) = O(\log n), \text{ According to fast power Algorithm.}$$

$$s = (k \cdot (h(m) + x \cdot r)^{-1}) \bmod q$$

$$O(s) = O((k \cdot (h(m) + x \cdot r)^{-1}) \bmod q)$$

$$O(s) = O(\log n), \text{ According to Extended Euclid's Algorithm.}$$

Therefore,

$$O(\text{signature}) = O(r) + O(s) = O(\log n) + O(\log n)$$

$$O(\text{signature}) = (2 \log n)$$

#### 4.6.10 McCurley Verification Complexity

This algorithm uses three steps to verify the signature, as follows:

$$u1 = (h(m) \cdot s) \bmod q$$

$$u2 = (s \cdot r) \bmod q$$

$$v = ((g^{u1} \cdot y^{u2}) \bmod p) \bmod q$$

$$O(u1) = O((h(m) \cdot s) \bmod q)$$

$$O(u1) = 1, \text{ One multiplication operation } h(m) \text{ and } s.$$

$$O(u_2) = O((s \cdot r) \bmod q)$$

$$O(u_2) = 1, \text{ One Multiplication of } s \text{ with } r.$$

$$O(v) = O\left(\left((g^{u_1} \cdot y^{u_2}) \bmod p\right) \bmod q\right)$$

$$O(v) = O(g^{u_1}) \cdot O(g^{u_2})$$

$$O(v) = O(\log n) + O(\log n)$$

$$O(v) = (2 \log n)$$

Therefore,

$$O(\text{verify}) = O(v) = O(2 \log n)$$

#### 4.6.11 Variant of DSA Signature Complexity

$$r = (g^k \bmod p) \bmod q$$

$$O(r) = O\left((g^k \bmod p) \bmod q\right)$$

$$O(r) = O(r) = (\log n)$$

$$s = \left(k \cdot (x(h(m) + r))^{-1}\right) \bmod q$$

$$O(s) = O\left(\left(k \cdot (x(h(m) + r))^{-1}\right) \bmod q\right)$$

$$O(s) = O(\log n), \text{ According to Extended Euclid's Algorithm.}$$

Therefore,

$$O(\text{signature}) = O(r) + O(s) = O(\log n) + O(\log n)$$

$$O(\text{signature}) = O(2 \log n)$$

#### 4.6.12 Variant of DSA Verification Complexity

This algorithm using three steps to verify signature as follows:

$$u1 = (h(m) + r) \bmod q$$

$$u2 = (s \cdot u1) \bmod q$$

$$v = (y^{u2} \bmod p) \bmod q$$

$$O(u1) = O((h(m) + r) \bmod q)$$

$$O(u1) = 1, \text{ One addition operation of } h(m) \text{ to } r.$$

$$O(u2) = O((s \cdot u1) \bmod q)$$

$$O(u2) = 1, \text{ One multiplication of } s \text{ with } u1.$$

$$O(v) = O((y^{u2} \bmod p) \bmod q)$$

$$O(y^{u2}) = O(\log n), \text{ According to fast power algorithm.}$$

Therefore,

$$O(\text{verify}) = O(v) = O(\log n)$$

The following Table summarizes the result of computations previous algorithm and compare their BigO complexity.

**Table(4.11) Summary of BigO Notation.**

No.	Algorithm	Avg. Signature time( $\mu$ sec)	Signature BigO	Avg. Verify time( $\mu$ sec)	Verification BigO	Total algorithms BigO
1	DSA	15165.7	$2\log n$	29092.6	$3\log n$	$5\log n$
2	GOST	13131.04	$\log n$	41062.09	$3\log n$	$4\log n$
3	McCurley	15242.13	$2\log n$	27255.33	$2\log n$	$4\log n$
4	Yen-Laih	15220.92	$2\log n$	29286.99	$3\log n$	$5\log n$
5	Variant of DSA	15181.53	$2\log n$	13493.13	$\log n$	$3\log n$
6	M.DSA	15443.97	$2\log n$	13644.08	$\log n$	$3\log n$

The result illustrated in Figure(8).

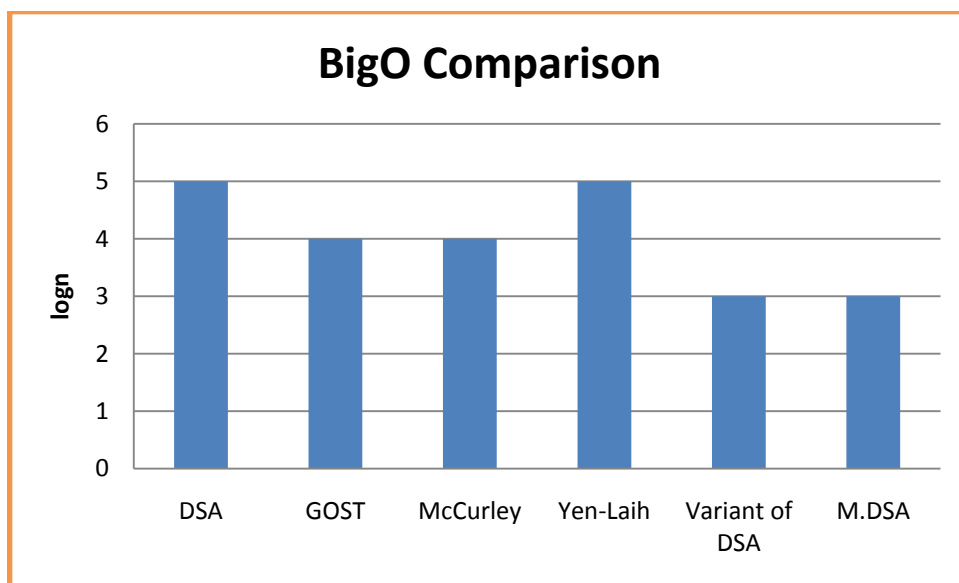


Figure (8) Comparison of Variants of DSA BigO

## 4.7 Number of Operations in Algorithm

In this section a comparison between six algorithms of their signature and verify processes in arithmetic operations is performed then the total number of arithmetic operations in each algorithm is calculated.

### 4.7.1 Signature Generating Process

Signing part of the algorithms is differ in each algorithm according to the modification of the original DSA, in general they algorithms use the same equation to calculate the part  $r$  of the signature but differ equation for calculating  $s$ , a summary of the arithmetic operations of this process is shown in table below.

**Table(4.12) Algorithm Number of Arithmetic Operations in Signature.**

No.	Algorithm	Mod/Inv.	Mod Exp.	Add/Sub.	Mult.	Modulus	Total
1	DSA	1	1	1	2	3	8
2	Yen-Laih	1	1	1	2	3	8
3	GOST	0	1	1	2	3	7
4	McCurley	1	1	1	2	2	8
5	Var. of DSA	1	1	1	2	2	7
6	M.DSA	1	1	1	2	2	7

### 4.7.2 Verification Process

The verification process got good improvement in total time and arithmetic operations, many algorithms try to remove the most time required factor which is the Inverse to gain speed in verification, the following table showing the arithmetic operations of the verification process for several algorithms compared with the M.DSA and original DSA.

**Table(4.13) Algorithm Number of Arithmetic Operations in Verification.**

No.	Algorithm	Mod/Inv.	Mod Exp.	Add/Sub.	Mult.	Modulus	Total
1	DSA	1	2	0	3	4	10
2	Yen-Laih	1	2	0	3	4	10
3	GOST	0	3	1	3	4	11
4	McCurley	0	2	0	3	3	8
5	Var. of DSA	0	1	1	1	3	6
6	M.DSA	0	1	0	2	3	6

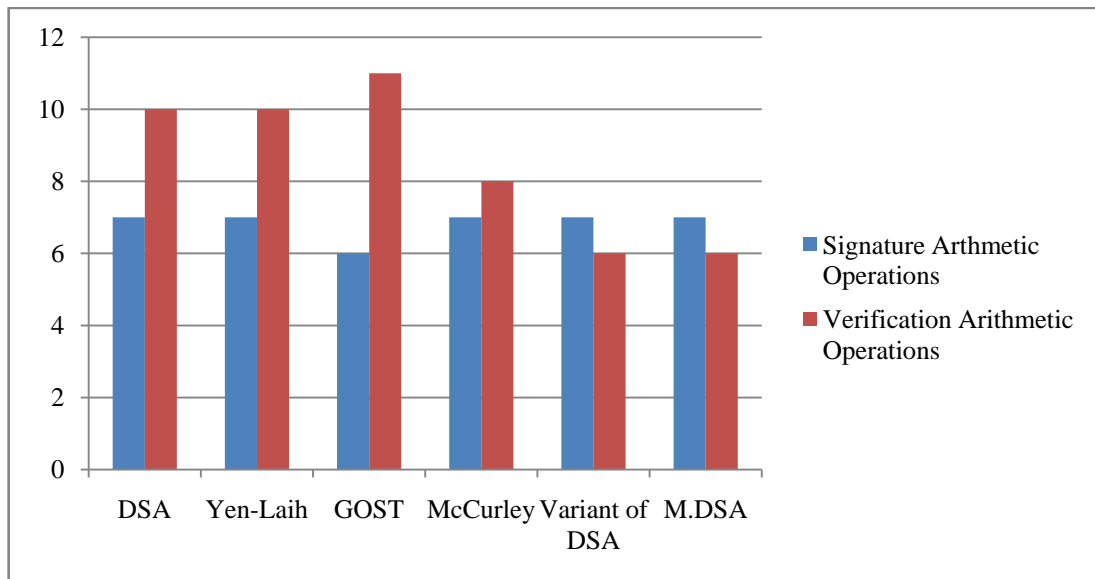


Figure (9) Comparison of Verify and Signature Operations



## **Chapter Five**

### **Conclusion and Future Work**

## Chapter Five

### Conclusion and Future Work

#### 5.1 Conclusion

This thesis aims to improve digital signature algorithm by decreasing the time complexity this achieved by reducing the calculation steps in the original DSA while keeping the same parameters strength. From the practical experimentation, the following conclusions may be drawn.

- Digital Signature use mechanism that provide authentication of the sender's identity and the integrity of the received data even if the communication channel is insecure.
- The proposed M.DSA has manifested short verification time as compared with the NIST-DSA. This achieved through the reduction in the processing steps.
- The comparison of signing time for M.DSA algorithm with other DSA variants, such as Yen-Laih, McCurley, GOST, and Var-DSA algorithms have shown almost all equal except GOST was faster. However, the verification time M.DSA gave superiority to all except that for Var-DSA was faster.
- The overall speed gains for both signing and verification have indicate a quite satisfying results.
- BigO calculations were included in the research to solidify the improvement obtained which means an improvement in its time complexity.

- It was noticed that the computation time fluctuates for changing the length of the private key and the generated random number, hence the average time for large number of trials is taken for all measurements. This indicates that private key and message random integer length do not affect the signature speed, but the procedure is the cause of the improvement in computation time.

## 5.2 Future Work

Future work on digital signature may be extended to the following problems:

- Include the use of RSA, and elliptic curve DSA for the time complexity measurements.
- Implement the proposed M.DSA algorithm in certain applications such as Blind signature, and Group signature. Applications that may be involved are commercial ones such as e-government, e-Banking, and e-election.
- Implement this technique on access control, personal identification systems and intrusion detection.
- It may be also beneficial to implement it in coordination with biometrics to generate signature.
- Due to the fast signing process for GOST Digital Signature technique and the obtained implement in the signature validation process, one an investigation is worthwhile to combine these two schemes in order to look for more improvement in DSA that may be achieved for signing and verification.

## References

- Abidi, A., Bouallegue, B., & Kahri, F. (2014). Implementation of elliptic curve digital signature algorithm (ECDSA). In *Computer & Information Technology (GSCIT), 2014 Global Summit on* (pp. 1-6). IEEE.
- Ali, H. A. (2004). Improved Verification Speed Enhancement for Digital Signature Using Discrete Algorithm Variant. *Journal of the Association of the Advancement of Modeling and Simulation Techniques in Enterprises (AMSE), Signal Processing and Pattern Recognition*, Vol. 47, No. 4, France, PP: 17-27.
- Alpizar-Chacon, I., & Chacon-Rivas, M. (2016). Authenticity and versioning of learning objects using the digital signature infrastructure of Costa Rica. In *Learning Objects and Technology (LACLO), Latin American Conference on* (pp. 1-6). IEEE.
- Andrade, E. R., & Terada, R. (2016). Proposal of Enhancement for Quartz Digital Signature. *Brazilian Journal of Information Security and Cryptography*, 2(1), 3-15.
- Bajaj, S. B., & Grewal, M. (2015). TL-SMD: Two layered secure message digest algorithm. In *Advance Computing Conference (IACC), 2015 IEEE International* (pp. 349-352). IEEE.
- Chang, X. (2009). PDFeH: A PDF Based Generic Teacher-Student E-Homework System. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on* (pp. 1-4). IEEE.
- Dhagat, R., & Joshi, P. (2016). New approach of user authentication using digital signature. In *Colossal Data Analysis and Networking (CDAN), Symposium on* (pp. 1-3). IEEE.
- Diffie, W., & Hellman, M. (1976). New directions in cryptography. *IEEE transactions on Information Theory*, 22(6), 644-654.
- ElGamal, T. (1984). A public key cryptosystem and a signature scheme based on discrete logarithms. In *Workshop on the Theory and Application of Cryptographic Techniques* (pp. 10-18). Springer Berlin Heidelberg.
- ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4), 469-472.
- Engström, Pontus (2016). "Modernizing forms at KTH: Using Digital Signatures."
- Fu, C., & Zhu, Z. L. (2008). An efficient implementation of RSA digital signature algorithm. In *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing* (pp. 1-4). IEEE.
- Gallagher, P., & Kerry, C. Fips pub 186-4: Digital signature standard, dss (2013).
- Halton, J. H., & Ribenboim, P. (1993). *The Little Book of Big Primes*.

- Hernandes, P. R. G., & Carvalho, L. F. (2014, November). Digital Signature of Network Segment Using Flow Analysis through Genetic Algorithm and ACO Metaheuristics. In *Chilean Computer Science Society (SCCC), 2014 33rd International Conference of the* (pp. 92-97). IEEE.
- Jaju, S. A., & Chowhan, S. S. (2015). A Modified RSA algorithm to enhance security for digital signature. In *Computing and Communication (IEMCON), 2015 International Conference and Workshop on* (pp. 1-5). IEEE.
- Manickam, S., & Kesavaraja, D. (2016). Secure multi server authentication system using elliptic curve digital signature. In *Circuit, Power and Computing Technologies (ICCPCT), 2016 International Conference on* (pp. 1-4). IEEE.
- Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of applied cryptography*. CRC press.
- Na, Z., & Xi, X. G. (2008). The application of a scheme of digital signature in electronic government. In *Computer Science and Software Engineering, 2008 International Conference on* (Vol. 3, pp. 618-621). IEEE.
- NIST.FIPS.186-4, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>.
- Poulakis, D. (2009). A variant of digital signature algorithm. *Designs, codes and cryptography*, 51(1), 99-104.
- Ramzan, Z. A. (1999). *Group blind digital signatures: Theory and applications* (Doctoral dissertation, Massachusetts Institute of Technology).
- Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2), 120-126.
- SadrHaghighi, S., & Khorsandi, S. (2016). An identity-based digital signature scheme to detect pollution attacks in intra-session network coding. In *Information Security and Cryptology (ISCISC), 2016 13th International Iranian Society of Cryptology Conference on* (pp. 7-12). IEEE.
- Schneier, B. (2000). *Applied Cryptography Second Edition: protocols, algorithms, and source*. Beijing: China Machine Press.
- Schnorr, C. P. (1989). Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology* (pp. 239-252). Springer New York.
- Shacham, H. (2005). *New paradigms in signature schemes* (Doctoral dissertation, Stanford University).
- Singh, M., Kaur, H., & Kakkar, A. (2015). Digital signature verification scheme for image authentication. In *2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS)* (pp. 1-5). IEEE.

Stallings, W. (2013). *Cryptography and Network Security: Principles and Practice*.

Tan, C. H., Yi, X., & Siew, C. K. (2003). Signature scheme based on composite discrete logarithm. In *Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on* (Vol. 3, pp. 1702-1706). IEEE.

Tan, Y., Tang, S., Chen, J., Yu, Y., & Li, X. (2016). Building a new secure variant of Rainbow signature scheme. *IET Information Security*, 10(2), 53-59.

Thangavel, J. (2014). Digital Signature Comparative study of its usage in developed and developing countries.

Zhang, Q., Li, Z., & Song, C. (2011). The Improvement of digital signature algorithm based on elliptic curve cryptography. In *Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 2011 2nd International Conference on* (pp. 1689-1691). IEEE.

## Appendix A

Table (4.2) Signing and Verification time calculation for DSA, with p, q and g of length of 100 digit, 10 values of k for each x value.

No.	X	k	Avg. sign time (μsec)	Avg. verify time (μsec)
20	6926258135772437824 5	151009910461393148438986	8388.7	56835
		3087878155320077479132		
		282997156268368481892711		
		221302006479518306171006		
		182275415273788523195640		
		375474902654144642730319		
		350057542284960735886137		
		218516958774789246472262		
		8370895920735987579790		
		237052374525416383903625		
40	1458119622525378455 4317430885832858578 03	151009910461393148438986	14803.8	67352.3
		6054003189351918842314407 34089662335		
		1205186176661931374958280 04346137354		
		2887015087795995523096034 44641091371		
		3191396913838816573111754 95030641033		
40	1458119622525378455 4317430885832858578 03	4779507177700238446474962 83748395202	14803.8	67352.3

		1498948315193195519945756 36847014036		
		5553719217226048907043011 19460752512		
		2946591721540529840679972 66527869523		
		4004008660110200754243455 37902287163		
		6389965031255953214410469 35695496481		



60	39582	649410958545021274958875574929828193079 989206544196604331389	19606.3	56644.5
	15979	709452709871085259872218555283606939794 701201954867673277189		
	06207	345060878198596119875743354713830892728 143994542484980799146		
	52726	146024624685714844460088060264960946321 700885841508125641303		
	60808	603654797411777991405738225439592784606 635524934305943520461		
	66483	192898918165157257522282216924365935410 910373979671379352032		
	48316	643659003252315467320539648822079159361 886472787612007285574		
	86396	268547026447605643072549705830106407053 682340224233814511845		
	86529	364921384021859268984238901082818300138 221327599575647361127		
	40934	115108226878501565302399315110447188938 893530649153555230840		

80	24493	585306849444757013682910036313488140749 148764019834270940176786626155397220500 015382	22509.4	50548.6
	07626	541705983142417210236053646414157587220 376751936412035428359606040362937317326 987912		
	74506	703986809512278202929053458533545184417 355142405961749404571672512260154468347 810594		
	96355	736402293099073922151995776668127376920 776454877848612248928496354261009090639 760229		
	33521	907516261285370230596753339540674868966 043560563037882670922856232131217692185 854639		
	98723	760341154867821556005582169845983879464 794894437504053246631773407514630048360 844092		
	87830	147789169324593709266089037578469459746 180647337166887317656712275075952894331 450401		
	26039	372647380571444510253108154466730372106 366303949972241209045771111932793285525 70818		
	79500	817686139652876225916833436544204056995 023207573757995691050863462174610198609 94120		
	85937	683034685463857216648527684318042854028 224223004544490381410784221440338190181 500191		



**Table (4.3) Signing and Verification time calculation for M.DSA, with p, q and g of length of 100 digit, 10 values of k for each x value.**

No.	x	k	Avg. sign time ( $\mu$ sec)	Avg. verify time ( $\mu$ sec)
<b>20</b>	8425275 6894206 064432	180769088745145988147593	8266.3	17475.6
		483634472128405933266674		
		121548062405120620638350		
		225177911597334692093774		
		513744180531264711199504		
		64725595382775582142328		
		79486900966789377996676		
		250191358658632140316280		
		313902445970086310861518		
		359661110658913110567079		
<b>40</b>	7400016 4874691 2785859 1431200 9724510 40017	627063601167163623662250751837879501 460786316498	16027.8	22117.8
		469947594552973778967789171389097835 19633342644		
		175625733212590064687402092603647060 983853150142		
		250902136547156762980234751229767274 513157893030		
		399514835656188779275037504134967313 950697555902		

		276713215821825442659015237981621564 209593776997		
		530512288304193155901465386045882538 533985021532		
		468263829227426449047200720674691755 006453952085		
		539769596960298651651590135825768430 606954769203		
		847015585045596228136118953342882881 28323763289		

<b>60</b>	9619142 7071445 7619712 3005887 8243238 5907047 1692300 1159720 5532	293427503231286766493764664974740005 476403386952851342729893	12726.7	16126.2
		783948731334027274175865902299437780 631980265658789681578851		
		624568143945174356267297090163108867 810007518562446557718023		
		717196128319306641363989443247391737 549756270501172024752983		
		171998289515939812828707035785001014 070048718089475883200109		
		431035594740398397113161881422473400 992482842286193395483180		
		371785097233720200811899256661202203 931219308427408446323520		
		709889810943171923280926368264908431 174345759053433358478880		
		140662394899518389066101460976784754 03298994554170922460529		
		773255989878138914977986054534277020 378130114792576244319291		

80	7417146 2359933 2541895 7223410 0065931 8369023 8399348 2619108 8851679 8237429 8374293 812	596727193833296999099697634698737397 767551578029364595831562	12181.1	15869.2
		218859146345005233217992254186001773 764521873021349445467469		
		677354351203471073417028701010107507 427740965298425552821985		
		250784953387341543743156161190545008 421653572583013047777350		
		504901297221472751100803012093608261 196937005849240869833572		
		531116049029159182540796333739655018 335819969376466265363961		
		246820554181824617965490805307287625 673706629133448104095888		
		383617333746805157737569000810534193 096983087825859850402465		
		634187988420460087364536617222831262 206245474832123859037409		
		777986447839425081206438155793810263 270876142311523445528688		

100	4801212 4007146 3177259 6191427 0714457 6197123 0058878 2432385 9070471 6923001 1597205 5092823 6628900 8310850 0	867617045222015540674089473361852843 545498437330992735018864526805763302 839707053171496162137920	18051.6	16169.3
		829835206840104422814868370541867581 445720609658031402578206689495441033 32532592627460024283827		
		491358537103754644211723479004437476 020236874217709062452299238093423045 988536747907293365539734		
		282683258292482688150327224181531435 927263735501806917244384213351465263 564030279517440690452782		
		257498656720085921080338971387276341 671670285375958823978725037417442597 696429404404053853931867		
		840016115159172089016896336026507392 657381285937874232215133696026906823 425507869081347783865488		
		102719692482185542177660530202793952 500382926489822257325679337659795578 7745939191592123868109492		
		101104829846980170789327556529850009 863527728568242528835001991161013371 6400493316423249474898603		
		391780274822045917130676947322453318 811398935000753340877606608977139102 296550955010633794091294		
		733253221399337357049103230000838153 101588254079345411426639077115712822 606820438335749925330511		