جـــامـعـة الـــشـرق الأوسـط
MIDDLE EAST UNIVERSITY
Amman - Jordan

Enhancement of Digital Signature Scheme

تعزيز نظام التوقيع الرقمي

Prepared by

Ruqa Abdulkareem Salih Al-Shnawa

Supervisor:

Prof. Hamza A. 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

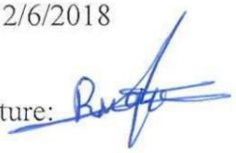
June, 2018

Authorization

I **Ruqa Abdulkareem Salih** Al-**Shnawa** authorize Middle East University to provide and electronic copies of my thesis to the libraries, organization, or bodies and institutions concerned in research and scientific studies upon request.

Name: Ruqa Abdulkareem Salih Al-Shnawa

Date: 2/6/2018

Signature: 

Decision of Discussion Committee

This thesis entitled "**Enhancement of Digital Signature Scheme** "has been discussed and accepted on: 2/6/2018

Members of Discussion Committee **Signature**

1- Chairman and Supervisor: Prof. Hamza Abbass Al-Sewadi

Professor, Department of Computer Information Systems

Middle East University (MEU)

2- Member: Prof. Abdallah Alashqur

Professor, Department of Computer Information Systems

Middle East University (MEU)

3- External Examiner: Dr. Abdulsalam W Al-Arabeyyat

Associate professor, Department of Computer Science

Al-Balqa Applied University (BAU)

Acknowledgment

(....رَبِّ أَوْزِعْنِي أَنْ أَشْكُرَ نِعْمَتَكَ الَّتِي أَنْعَمْتَ عَلَيَّ وعَلَىٰ وَالِدَيَّ وَأَنْ أَعْمَلَ صَالِحًا تَرْضَاهُ وَأَصْلِحْ لِي فِي ذُرِّيَّتِي إِنِّي تُبْتُ إِلَيْكَ وَإِنِّي مِنَ الْمُسْلِمِينَ) (15) { سورة الأحقاف }.

All praise and thanks are due to the Almighty Allah who always guides me to the right path and has helped me to complete this thesis. There are many people whom I have to acknowledge for their support, help and encouragement during the journey of preparing this thesis. So, I will attempt to give them their due here, and I sincerely apologize for any omissions.

First and foremost, I would like to record my heartfelt gratitude to my supervisor Prof. Hamza Abbass Al-Sewadi for his skilful supervision, advice and patient guidance throughout the work. Above all and the most needed, he provided me unflinching encouragement and support in various ways. I am really indebted to him more than he knows.

I would also like to express my deepest gratitude to all the respectable lecturers at the Faculty of Information Technology, Middle East University and special thanks to the Graduate School Office and the Library Staff and for everyone who supports me especially my colleague Mohammed Mustafa Rifaat to accomplish this work.

The Researcher

Dedication

To:

My parents who have always loved me unconditionally and whose good examples have taught me to work hard for the things that I aspire to achieve.

My beloved brother and sisters for their love and kindness, endless support and encouragement.

My lovely husband who has been a constant source of support and encouragement.

My daughter (Asl), the spirit of my life.

Everyone gives me support.

I dedicate this thesis.

The Researcher

Table of Contents

List of Tables

List of Figures

List of Appendixes

Table of Abbreviations

Abbreviations	Meaning
ANS	American National Standard
CA	Certification Authority
DL	Discrete Logarithm
DLP	Discrete Logarithm Problem
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
ECC	Elliptic Curve Cryptosystems
ECDLP	Elliptic Curve Discrete Logarithm Problem
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
H	Hash Function
K_p	Public key
K_r	Private key
M	Message
M.DSA	Modified Digital Signature Algorithm
NIST	National Institute of Standards and Technology
P	Plaintext
PKCS	Public Key Cryptography Standard
PKI	Public Key Infrastructure
RGB	Random Bit Generator
RSA	Algorithm developed by Rivest, Shamir and Adleman
SHA	Secure Hash Algorithm
TTP	Trusted Third Party

Enhancement of Digital Signature Scheme

Prepared by

Ruqa Abdulkareem Salih Al-shnawa

Supervisor

Prof. Hamza A. Al-Sewadi

Abstract

A digital signature is a way to ensure the authenticity of the data source or messages received within the field of the digital world that is equivalent to the traditional handwritten signature used for classical authentication. In recent years, a number of digital signature algorithms were developed and used such as DSA, RSA, ElGamal, GOST, etc. together with many of their variants. They vary in signing and verification execution speed, some have fast signing speed, while others have fast verification speed. Comparatively, GOST digital signature algorithm has the shortest signing time but longest verification time, and hence an improvement in its signature verification time is sought.

This thesis presents a modified version for GOST digital signature algorithm, called M.GOST. Its main objective is to improve the signature verification speed of this algorithm by reducing the computation complexity and benefit from its efficient signing speed. The authentication is achieved by reducing the calculation steps in the original GOST while preserving the strength of the parameters themselves. This thesis also contains the mathematical proof of this modified algorithm.

An investigation of the original GOST algorithm is performed first, then the suggested modified GOST variants (M.GOST) is tested for various parameter values. The time complexity is also compared with those of other available digital algorithms. The results of the comparison indicate that the proposed model achieved an improvement of about one and a half times faster signature verification speed over the original algorithm, using the same values for the general parameters, public and private key, random numbers, etc. for both signing and verification processes. Therefore, it is recommended to use the suggested version of the algorithm in applications that require short time for both, signing and verification.

Keywords: digital signature algorithms, authenticity, NIST-DSA, DSA variants, signature verification time complexity, cryptography, discrete logarithms.

تعزيز نظام التوقيع الرقمي

إعداد

رقى عبد الكريم صالح الشناوة

إشراف

الأستاذ الدكتور حمزة عباس السوادي

المُلخص

التوقيع الرقمي هو طريقة للتأكد من وثوقية مصدر البيانات او الرسائل المستلمة ضمن مجال العالم الرقمي حيثأنه يعتبر مكافئ للتوقيع التقليدي المكتوب باليد والذي يُعبّر عن مصدر هذه المستندات وتوثيقها، وقد طورت حديثاً العديد من خوارزميات التوقيع الرقمي مثل DSA وRSA وElGamal وGOST وغيرها. اضافة الى التعديلات عليها. وهي تتفاوت في سرعة تنفيذ التوقيع والتحقق منه. فمنها السريعة في عملية التوقيع واخرى سريعة في التحقق من التوقيع.

وبالمقارنة فان خوارزمية التوقيع الرقمي GOST هي الاسرع في التوقيع ولكنها الأبطئ في عملية التحقق من التوقيع، لذا فان التحسين في زمن التحقق من التوقيع هو المطلوب.

تقدم هذه الاطروحة خوارزمية معدلة من خوارزمية التوقيع الرقمي GOST أُطلق عليها اسم M.GOST حيث هدفها الرئيسي هو تحسين تعقيد خوارزمية التوقيع الرقمي من خلال تقليل تعقيد الوقت وتعقيد التحقق من هوية المرسل الذي تم تحقيقه عن طريق تقليل خطوات الحساب في GOST الأصلي مع الحفاظ على قوة المعلمات نفسها كما تحتوي هذه الاطروحة على الإثبات الرياضي لهذه الخوارزمية المعدلة.

تبدأ الرسالة بإجراء دراسة لخوارزمية التوقيع الرقمي GOST الاصلية و كذلك عدة خوارزميات للتوقيع الرقمي وإصدارات مختلفة منها لإختبار تأثير مُعاملات التوقيع الرقمي على إداء هذه الخوارزميات حيث تم حساب و مقارنة معدل وقت التوقيع و التحقق منه لأطوال مختلفة لمعاملات خوارزمية التوقيع الرقمي الاصلية، و نتيجة المقارنة تشير الى تفوق النموذج المُقترح لمعامل المفتاح الخاص و الرقم الاولي العشوائي من ناحيتي وقت التوقيع ووقت التحقق من التوقيع ، و كذلك التعقيد العام للخوارزمية المعدلة على بقية الخوارزميات حيث كانت اسرع بحوالي مرة ونصف من الخوارزمية الاصلية بشكل عام. ومن هنا يوصى بإستخدام الإصدار المُقترح من الخوارزمية في التطبيقات التي تحتاج سرعة في حساب وقت التوقيع ووقت التحقق من التوقيع معاً.

الكلمات المفتاحية: خوارزميات التوقيع الرقمي، الوثوقية، NIST-DSA، تطويرات DSA، زمن التحقق من التوقيع، التشفير، اللوغارتيمات المتقطعة.

Chapter One

Background of the Study and the study Importance

Chapter One

Background of the Study and the study Importance

1.1. Introduction

A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document.

The digital equivalent of a handwritten signature or stamped seal, but offers far more inherent security, a digital signature is intended to solve the problem of tampering and impersonation in digital communications. Digital signatures can provide the added assurances of evidence to origin, identity and status of an electronic document, transaction or message, as well as acknowledging informed consent by the signer.

In many countries, including the United States, digital signatures have the same legal significance as the more traditional forms of signed documents. The United States Government Printing Office publishes electronic versions of the budget, public and private laws, and congressional bills with digital signatures (Paul, 2017).

Digital Signature Standard (DSS) was defined by the National Institute of Standards and Technology (NIST) to be used for senders' authentication and message integrity. Hence, it is used for signing and verification of messages as well as assuring their integrity.

Digital Signature Algorithm (DSA) is specified and referred to as NIST-DSA. The specification includes criteria for the generation of domain parameters, for the generation of public and private key pairs, and for the generation and verification of digital signatures.

This Standard includes requirements for obtaining the assurances necessary for valid digital signatures. Methods for obtaining these assurances are provided in NIST Special Publication (SP) 800-89, Recommendation for Obtaining Assurances for Digital Signature Applications (Merkle,1989).

The efficiency of DSA for some sensitive application is highly required. Hence signing and verification time are required to be as short as possible. Many varieties of DSA have been successful in improving either signing time or verification time. For example, GOST variant achieved improvement on the signing time while Yen-laih, McCurley, Ali, and Naccache variants achieved improvement on the verification side. An improvement in both signing and verification is sought and highly demanded in many sensitive data transfer and management applications.

This work proposes a technique that look for improvement in the signing and verification time complexity by mixing the policy of GOST variant for signing digital document with any of the other variants that does the signature verification faster than DSA. Therefore, processing time would be reduced for both signing and verification of the signature.

1.2. Keywords Definitions (Kerry, Gallagher 2013)

It is important at the beginning to clarify the exact meaning of the common terms used in the field of the study. as briefly defined in the following:

Assurance of domain parameter validity: Confidence that the domain parameters are arithmetically correct.

Assurance of possession: Confidence that an entity possesses a private key and any associated keying material.

Assurance of public key validity: Confidence that the public key is arithmetically correct.

Certificate: A set of data that uniquely identifies a key pair and an owner that is authorized to use the key pair. The certificate contains the owner's public key and possibly other information, and is digitally signed by a Certification Authority (i.e., a trusted party), thereby binding the public key to the owner.

Digital signature: The result of a cryptographic transformation of data that, when properly implemented, provides a mechanism for verifying origin authentication, data integrity and signatory non-repudiation

Domain parameters: Parameters used with cryptographic algorithms that are usually common to a domain of users. A DSA or ECDSA cryptographic key pair is associated with a specific set of domain parameters.

Hash function: A function that maps a bit string of arbitrary length to a fixed length bit string. Approved hash functions are specified in FIPS 180 and are designed to satisfy the following properties:

One-way: It is computationally infeasible to find any input that maps to any new pre-specified output, and

Collision resistant: It is computationally infeasible to find any two distinct inputs that map to the same output.

Hash value: message digest.

Intended signatory: An entity that intends to generate digital signatures in the future.

Key: A parameter used in conjunction with a cryptographic algorithm that determines its operation. Examples applicable to this Standard include:

The computation of a digital signature from data, and

The verification of a digital signature

Key pair: A public key and its corresponding private key.

Message: The data that is signed. Also known as "signed data" during the signature verification and validation process.

Security strength: A number associated with the amount of work, or the number of operations, required to break a cryptographic algorithm or system. Sometimes referred to as a security level.

Signature validation: The mathematical verification of the digital signature and obtaining the appropriate assurances.

Signature verification: The process of using a digital signature algorithm and a public key to verify a digital signature of data.

1.3. Basic of Digital Signature Process

Digital signatures are based on public key cryptography which is also known as asymmetric cryptography. Using a public key algorithm such as Digital Signature Algorithm (DSA), Rivest-Shamir-Adleman (RSA), El Gamal, and many more, one can generate two keys or more that are mathematically linked: some private and others are public. To create a digital signature, (signing a document) a one-way hash value of the digital data to be signed is created.

The private keys are then used to encrypt the hash. producing the document signature. This signature along with other information, such as the hashing algorithm and signer public key are sent to the recipient as shown in Figure 1.1(Atreya, Hammond, Paine, Starrett, & Wu:2002). The reason for encrypting the hash instead of the entire message or document is that a hash function can convert an arbitrary input into a fixed length value, which is usually much shorter. This saves

time since hashing is much faster than signing. The value of the hash is unique to the hashed data. Any change in the data, even changing or deleting a single character, results in a different value.

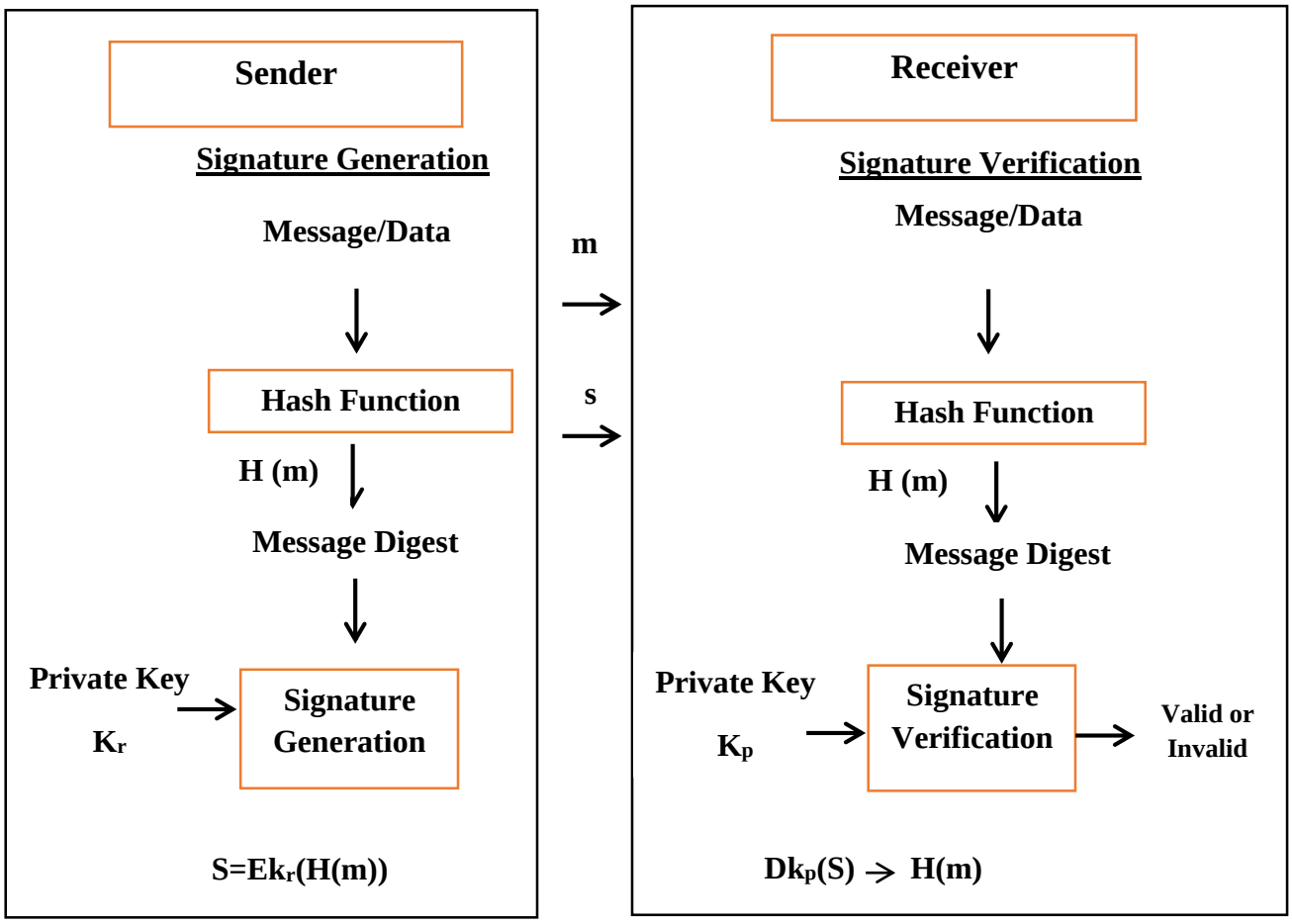


Figure 1.1 Digital Signature Processes (Prepared by the researcher)

To sign a digital message **m**, it is first hashed by the hash function **H** producing the hash value **H(m)**, then this value is signed with private key $\mathbf{K_r}$ and sent over to the recipient together with the message and the senders public key $\mathbf{K_p}$.

At the receiver side, the received signature **S** together with public keys $\mathbf{K_p}$ and the message **m** can verify the signature after decrypting the signature to get the hash value which will be compared with that produced by hashing the message **m**.

If the two hash values match, the message has not been tampered with, and the receiver verifies the senders signature. But if the two hashes do not match the data has either been tempered with in some way (integrity) or the signature is not valid (i.e. the signer is not authentic). These processes will be explained in details in chapter 2.

1.4. Problem Statement

- The most widely used algorithm for digital signature is the NIST-DSA, that was adapted by National Institute of Standards and Technology. However, DSA has considerably long signing and verification time, so many DSA variants were developed, resulting in execution time improvement either on the signing side or on the verification side, such as Yen-Leih, McCurly, Modified DSA (M.DSA), GOST, etc.
- The GOST algorithm has the shortest signing time (but long verification time), while some DSA variants has the shortest Verification time, hence improvement in both sides is sought and will be problem tackled in this thesis.

1.5. Questions of the Study

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

1. What are the possible improvement and the time complexity of GOST algorithm when different equation configuration is adopted?
2. What are the effects of various parameters on the behavior of the modified digital signature algorithm (M.GOST)?
3. What is the effect of the private key length for different messages on the signing and verification time measurement?
4. Is there any effect of the length of message secret random integer on the signature generating process speed?

1.6. Objectives of the Study

The computations of digital signature algorithm generally rely on the choice of large primes. The use of discrete mathematics, involves multiplications and exponentiations of large numbers, and their security relies on the difficulty of analysis 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 time complexity of standard DSA, GOST, and (M.DSA) variant then check the effect of important factors on the time measurements for signing and verification.
2. Propose, design, and test a new modified version of the digital signature algorithm which involve both GOST and M.DSA modifications of the standard DSA.
3. Achieve time complexity improvements on both signing and validation sides.

1.7. Motivation

1. 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 improve both signing and verification times, such improvement will be useful for sender and verifier for application where time is crucial.
2. Improvement of computation complexity by altering the mathematical processes such as multiplication and exponentiation. Such modifications will certainly change the time needed for the signing and validation process that will affect the efficiency.

Chapter Two

Theoretical Background and Literature Review

Chapter Two
Theoretical Background and Literature Review

2.1. Introduction

A brief but comprehensive theoretical background will be described first in this chapter. It covers the definition of digital signature concept, signing, and verification, types of digital signature and the standard DSA. Then, a literature survey of DSA variants and other digital signatures algorithms will be presented.

2.2. Theoretical Background

A digital signature is an electronic analogue of a written signature; It is used to provide assurance that the claimed signatory truly signed the information. In addition, a digital signature may be used to detect whether or not the information was modified after it was signed (i.e., to detect the integrity of the signed data). These assurances may be obtained whether the data was received in transmission or retrieved from storage. A properly implemented digital signature algorithm that meets the requirements of this Standard can provide these services (Chang, 2009).

Digital signature algorithm includes a signature generation process and a signature verification process. A signatory uses the generation process to generate a digital signature on data; a verifier uses the verification process to verify the authenticity of the signature (Merkle,1989).

Each signatory has a public and private key and is the owner of that key pair; the private key is used in the signature generation process. Where the key pair owner is the only entity that is authorized to use the private key to generate digital signatures, while the public key is used in the signature verification process (Atreya, Hammond, Paine, Starrett, & Wu, 2002).

The public key need not be kept secret, but its integrity must be maintained, as anyone may use it to verify a correctly signed message. Typically, for both the signature generation and verification processes, the message is converted into a fixed-length data string using an approved hash function. Both the original message and the digital signature are made available to a verifier, who already also knows the same hash function and the signatory's public key.

2.2.1. Use of Digital Signature

2.2.1.1. Integrity and Authentication

This attribute enables others to validate the integrity of the data by using the signer's public key to decrypt the hash. If the decrypted hash function matches a second computed hash of the same data, it proves that the data hasn’t been changed since it was signed. If the two hashes don't match, the data has either been tampered with in some way (Integrity) or the signature was created with a private key that doesn't correspond to the public key presented by the signer (authentication).

A digital signature can be used with any kind of message (whether it is encrypted or not), so the receiver can be sure of the sender's identity and that the message arrived intact.

2.2.1.2. Nonrepudiation

Digital signatures make it difficult for the signer to deny having signed something (non-repudiation) (assuming their private key has not been compromised) as the digital signature is unique to both the document and the signer, and it binds them together.

A digital certificate, is an electronic document that contains the digital signature of the certificate-issuing authority, binds together a public key with an identity and can be used to verify a public key belongs to a particular person or entity.

2.3. Digital Signature Generation and Verification

2.3.1. Digital Signature Generation

The processes of digital signature generation and verification will be outlined in more details in the following.

Figure 2.3 illustrates a typical block diagram for signing a digital message **(m)**. Prior to the generation of a digital signature for the message, it's digest shall be generated using an appropriate approved hash function. Then, the obtained hash value is encrypted (signed) by certain digital signature algorithm. Depending on the digital signature algorithm to be used, some additional information shall be obtained. For example, for DSA algorithm a random secret number per-message shall be generated. Using the selected digital signature algorithm, the signature private key, the message digest, and any other information required by the digital signature process, a digital signature shall be generated in accordance with this Standard (Menezes, van Oorschot and Vanstone, 1996).

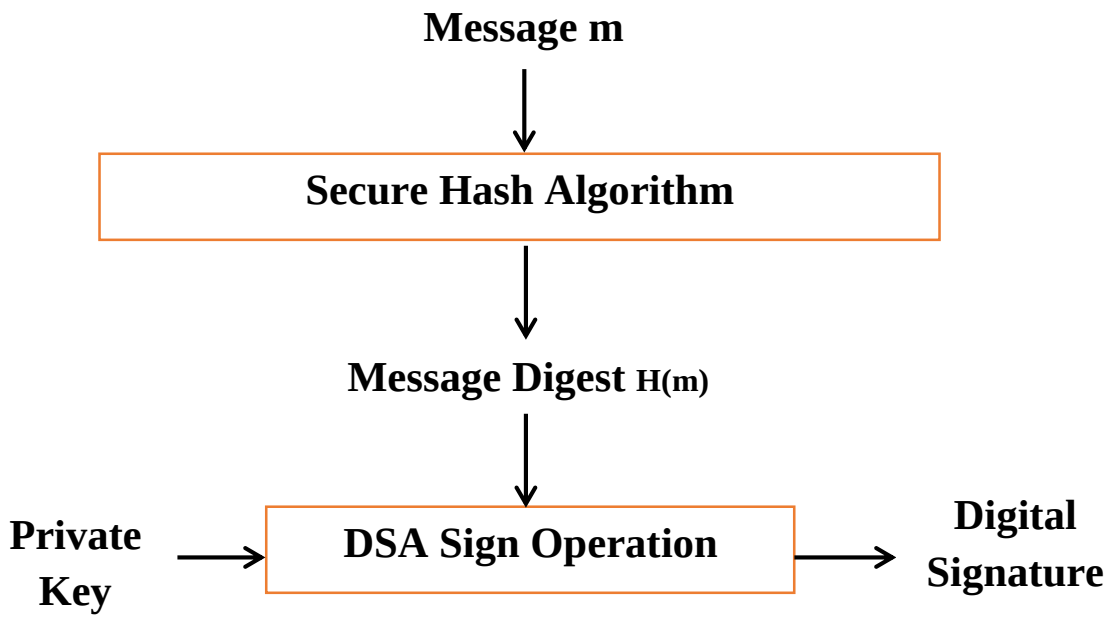


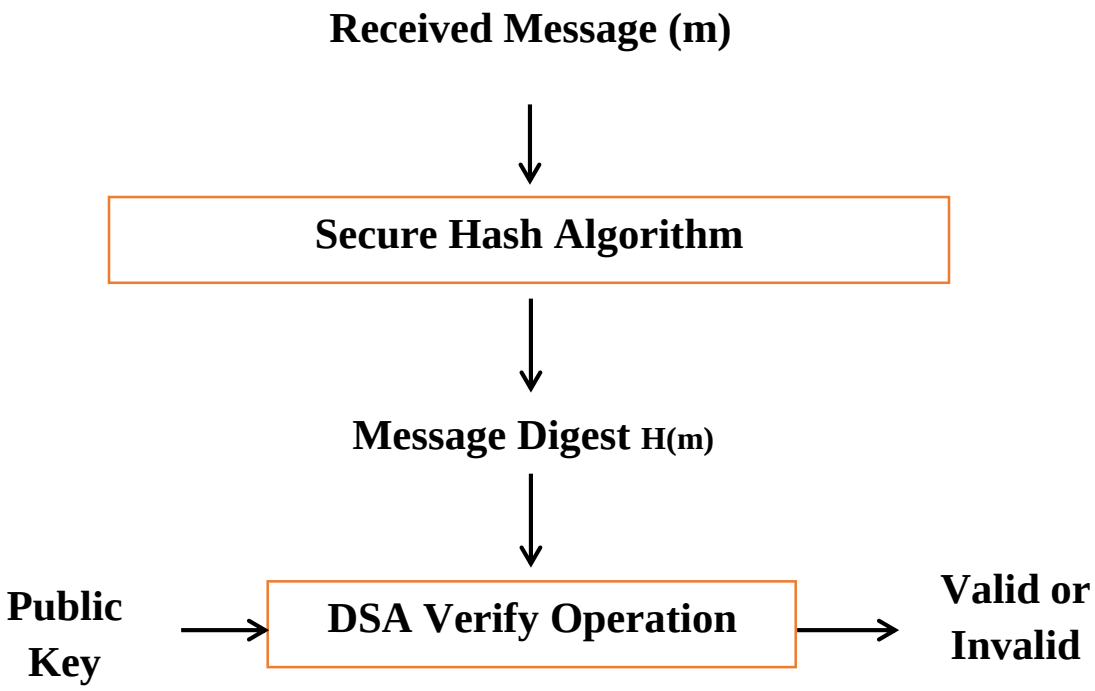
Figure 2.3 Digital Signature Generation

The signatory may optionally verify the digital signature using the signature verification process and the associated public key. This optional verification serves as a final check to detect otherwise undetected signature generation computation errors; this verification may be prudent when signing a high-value message, when multiple users are expected to verify the signature, or if the verifier will be verifying the signature at a much later time (Schneier, B. 2000).

2.3.2. Digital Signature Verification or Validation

Figure 2.4 depicts the digital signature verification and validation process that are performed by a verifier (e.g., the intended recipient of the signed data and associated digital signature). In order to verify a digital signature, the verifier shall obtain the public key of the claimed signer, (usually) based on the claimed identity. A message digest shall be generated on the message whose signature is to be verified using the same hash function that was used during the digital signature generation process.

Then, using the appropriate digital signature algorithm, the domain parameters (if appropriate), the public key and the newly computed message digest, the received digital signature is verified in accordance with this Standard. If the verification process fails, no inference can be made as to whether only the data is correct, or the sender is authentic. It can only be said that the signature is not validated.



Valid – Signature Verified

Or

Invalid – Signature Verification Failed

Figure 2.4 Digital Signature Verification or Validation

If DSA algorithm has been used to generate the digital signature, the verifier also obtains the domain parameters which are usually obtained, together with the public key from a certificate created by a trusted party or directly from the claimed signatory.

Before accepting the verified digital signature as valid, the verifier shall have

(1) Assurance of the signatory's claimed identity,

(2) Assurance of the validity of the domain parameters (for DSA and ECDSA),

(3) Assurance of the validity of the public key, and

(4) Assurance that the claimed signatory actually possessed the private key that was used

to generate the digital signature at the time that the signature was generated.

(IEEE Potentials, Volume: 25, Issue: 2, March-April 2006)

Methods for the verifier to obtain these assurances are provided in SP 800-89. Note that assurance of domain parameter validity may have been obtained during initial setup. If the verification and assurance processes are successful, the digital signature and signed data shall be considered valid. However, if a verification or assurance process fails, the digital signature should be considered invalid (Stallings, 2006).

2.4 Types of Digital Signature

Historically, public key implementation for digital signature started by two persons in 1976 (Whitfield Diffie and Artin E. Hellman at Stanford University in United State of America). They introduced a new way to exchange keys for more secure communication and a new method to generate and share cryptographic keys, that later had led to the invention of the algorithm, termed

asymmetric encryption/decryption algorithms as compared to the traditional symmetric system (Diffie and Hellman (D-H), 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.

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 former because it is slow for digital signature. Later on other digital signature algorithms were developed, namely ElGamal algorithm (ElGamal, 1985), Schnorr algorithm (Schnorr, 1989), and the digital standard algorithm (DSA) by NIST 1991 which last revision at 2013 (Kerry, Gallagher, 2013)**.**

The Digital Signature Algorithm (DSA) was specified in a U.S. Government Federal Information Processing Standard (FIPS) called the Digital Signature Standard (DSS).

Its security is based on the computational intractability of the discrete logarithm problem (DLP) in prime-order subgroups of Z_p^*. Elliptic curve cryptosystems (ECC) were invented by Neal Koblitz and Victor Miller in 1985.

They can be viewed as elliptic curve analogues of the older discrete logarithm (DL) cryptosystems in which the subgroup of Z_p^* is replaced by the group of points on an elliptic curve over a finite field. The mathematical basis for the security of elliptic curve cryptosystems is the computational intractability of the elliptic curve discrete logarithm problem (ECDLP). Since the ECDLP appears to be significantly harder than the DLP, the strength-per-key-bit is substantially greater in elliptic curve systems than in conventional discrete logarithm systems. Thus, smaller parameters, but with equivalent levels of security, can be used with ECC than with DL systems. However, ECDSA is outside the scope of this thesis, and only DSA and its modified versions will be considered here after.

There are many modifications for the DSA algorithm such as GOST, Yen-laih, McCurly, and other variants, they all sought improvement in the signing and verification efficiency and will be discussed and compared in the following:

2.5. Typical Digital Signature Algorithms

2.5.1. Diffie-Hellman (D-H)

In 1979, Diffie and Hellman have suggested a very well-known public-key distribution technique which is based on their exchanged public keys. It is a key agreement protocol establishes secret communication key(s) among all parties and based on the discrete logarithm problem that enables two parties to establish a common secret key. Nevertheless, this scheme did not present authentication instrument for the exchanged public keys. So, for attaining key authentication, in 1993, Arazi proposed replacing the message in the DSA algorithm with D-H exchange key. Afterwards, Nyberg and Rueppel have showed Arazi's scheme weakness stating that if one secret session key which is called as known key attack is compromised, then others will be also revealed. Later, Arazi's approach is expanded to firmly incorporate the D-H key exchange into the DSA which is called as secure D-H + DSA (Li Xin: 2007). D-H technique also suffers from what is known as man-in-the-middle attack weakness.

2.5.2 ElGamal

A signature elements of the digital signature is computed by first covering the long-term private key d utilizing a single additive process to join the key d with a first value in order to generate a digital signature of a message m. Then, the concealed value is multiplied by a second value to gain components (ElGamal,1985).

The first value is enumerated using the message m and another component of the digital signature, and the second value is obtained deploying the inverse of a component of the first value. In such a way, the signature component s is therefore produced using a process that counters the efficiency of side channel attacks, for instance, differential side channel analysis, by keeping away from a direct multiplication employing long-term private key d.

The above algorithm is in connection to D-H algorithm, where both the use of exponentiation in a finite field, and its security which is based on the rigidity of calculating discrete logarithms are the backbone of it. ElGamal algorithm's benefit is that each time exactly similar plaintext is encrypted. His algorithm provides different cipher text with one drawback where the cipher text length is double the length of the plain text. It also decrypts (verifies) the signature by employing private key to encrypt (sign) and public key (Menezes, van Oorschot and Vanstone, 1996). Thus, we can sum up ElGamal signature scheme as follows:

By looking at a cryptographic system having cryptographic parameters that involve a proper first number p and a generator α. A signee A has long-term private key d and public key., which is calculated by $y = \alpha^d \bmod p$.

Then, to generate an ElGamal signature for a message m, the following steps are used.

1- Select random integer k, with the value in the range 1 to p-2, where gcd (k, p−1) =1.

2- Calculate $r = \alpha^k \bmod p$ (2.1)

3- Calculate $s = k^{-1}(H(m) - d.r) \bmod (p-1)$ (2.2)

Whereas H(m) is the message hash function.

Thus, the generated signature is the pair (r, s). Whereas s must not be zero. Now both r and s are the signature.

In order to verify the signature, the following calculations are performed:

Compute, $v_1 \equiv r^s . y^r mod p.$ (2.3)

and $v_2 \equiv g^{H(m)} mod p.$ (2.4)

Then if $v_1 = v_2$, the signature is accepted as authentic.

In Cryptographic systems the setting information, electromagnetic emissions, power usage, or channel information are used to attempt and decide a secret value employed by the cryptographic unit during computing processes. Thus, the systems might be an issue to side channel attacks. As a result, multiplication in a computational unit of a cryptographic system is absolutely executed using a sequence of additions (ElGamal,1984). Therefore, through employing the side channel attacks, interlopers usually having enough awareness that in producing ElGamal signatures and their variants the long-term private key d is only used in one step of the generation of the signature which is in the calculation of the signature component s by way of the signing equation. Consequently, an interloper may seek using differential side channel analysis for gaining information about long-term private key d. In other words, an interloper would try to get

information from the side channel over the course of signing multiple messages comparing the differences between this information for acquiring information about private key d. It may also be feasible to remove enough information about long-term private key d for compromising its secrecy through examining those differences between the information upon multiple uses of private key d (i.e. upon multiple signing operations). Menezes, van Oorschot and Vanstone, (1996) stated that differential side channel analysis may compromise private key d with a greater probability if more processes in which long-term private key d is directly utilized in each signing operation.

2.5.3. Rivest-Shamir-Adleman (RSA)

RSA is a public-key cryptosystem developed by Ron Rivest, Adi Shamir, and Leonard Adleman. RSA involves the use of static keys, whereas the D-H key exchange algorithm required the dynamic exchange of keys. The RSA system reduces communications overhead with the ability to have static, unchanging keys for each receiver that are 'advertised' by a formal 'trusted authority' (the hierarchical model) or distributed in an informal 'web of trust'. The computational problem that RSA addresses the integer factorization problem. For example, a simple factorization problem is: What are the factors of the number 147? After trying a variety of numbers, such as 2, 4 and 5, it is evident that they will not divide equally into 147.

It soon becomes apparent that after experimenting with more numbers, 147 only has 3 and 7 as factors. This example is very easy, and it is worth remembering that Rivest, Shamir, and Adleman experimented with much larger numbers, numbers with over 100 digits. Subsequently, Rivest discovered a method that provides secure communications and did not suffer the key distribution problem. It can be used to encrypt messages and provide digital signatures. It is the most commonly used asymmetric algorithm, with high level of security (Alan Dhillon: 2002).

To sign a message m by RSA algorithm, the private key [d, N] is used in the equation.

$$S \equiv m^d mod N \quad (2.5)$$

To verify the signature, the public key [e, N] is used in the equation.

$$m \equiv S^e mod N \quad (2.6)$$

Where N is the product of two large primes p and q.

If the message m = 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 for the message rather than the message itself.

2.5.4. Digital Signature Algorithm (DSA)

As mentioned above, the Digital Signature Algorithm (DSA) is one of the variations of ElGamal digital signature scheme. In this algorithm, a signatory, which has public and private keys, is used to generate a digital signature of digital message; and a verifier to validate the authenticity of the signature as well. The private key of the signatory is utilized in the signature generation process whereas the public key is used in the signature verification process. For both signature generation and verification, the data (which is known as a message) is decreased by means of the Secure Hash Algorithm, like SHA which is identified in FIPS 180-1. Thus, the correct signature of the signatory cannot be generated if an adversary does not know the private key of the signatory. For more clarification, these signatures cannot be faked, however anyone can justify a correctly signed message by using the signatory's public key.

The verifier shall also gain the domain parameters in case the DSA is utilized for generating the digital signature. These public key and domain parameters may be concerted between the two communicating parties or gained from a trusted party (e.g., Certificate Authority, CA) (Atreya, Hammond, Paine, Starrett, & Wu: 2002).

To sign a message m digitally, the below equations are run to generate the signature, r and s:

$$r = (g^k \bmod p) \bmod q \quad (2.7)$$

$$s = (k^{-1}H(m) + xr) \bmod q \quad (2.8)$$

Whereas p, q, g, y are public parameters, which is long-term private key; k is a random integer for each message.

For verification purpose, the following is performed:

$$w = s'^{-1} \bmod q \quad (2.9)$$

$$v = ((g^{(H(m').w) \bmod q}.y^{(r'.w \bmod q)} \bmod p) \bmod q \quad (2.10)$$

If v = r then the signature is verified.

where r', s', m' are the received signature and message.

2.5.5. Short Comparison

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

Table (2.1) Comparison of Listed Algorithms

Algorithm	Advantage	Disadvantage
GOST	• Strong due to the change of random number (k), so, every time new cipher text is produced when the same plain text encrypted. • Signing time is shorter than DSA and variants.	• Long verification time compared with other digital signature algorithms
D-H *	• Challenging to find solutions for discrete logarithms. • No transmission of the shared key transmitted over the channel.	• Require an expensive exponential operation. • Utilized for setting up a secret key only rather than to encrypt or sign a message
RSA *	• Signing (or verification) process requires single equation. • Calculating Key generator does needs plenty of calculations.	• Operating slower than other symmetric cryptosystems.
ElGamal *	• Strong due to the change of random number (k), so, every time new cipher text is produced when the same plan text encrypted.	• Require randomness and slow-moving. • Cipher text length is double than the plan text.
DSA *	• Signature length does not depend on the message length. • Strong due to change of random key for each message	• Probability of verification fail because of S-1mod q if S=0 • Long verification time compared with other digital signature algorithms

* (M. Rifaat, 2017)

2.5.6. Modified DSA Algorithm (M.DSA)

There are various customized versions of standard DSA algorithm that are supported by the NIST, have been built up which ensured efficiency of the execution time measurement of either on the signing side or the verification side. The M.DSA, which was developed lately has shown good improvement in the verification time, and it will be included and compared with the proposed GOST algorithm in later chapter for examination of execution time for signing and signature verification.

In M.DSA versions, the equations contents of both signature and verification were altered. The computation of signature **s** is adjusted at the sender side, while one equation of the DSA verification calculation is deleted and the verification equation is also adjusted to accomplish the signature validation at the received side. These adjustments have decreased the verification time but reserved the same difficulty level for the signature and verification of NIST-DSA (M. Rifaat: 2017 and Ali, 2004). The M.DSA signing process consists of performing the following calculations

$$r = (g^k \bmod p) \bmod q \tag{2.11}$$

And $$s = (k.x.H(m) + r)^{-1} \bmod q \tag{2.12}$$

But only s is modified. It is noted that the value of r is the same as in NIST-DSA. For the verification process, the following equations are used instead of those for NIST-DSA.

$$u1 = (H(m) + r) mod q \tag{2.13}$$

$$u2 = (s.u1) mod q \tag{2.14}$$

$$v = (y^{u2}\, mod p) mod q \tag{2.15}$$

Then if the value of *v* is equal to the received r, the signature is verified, but if they do not have the same value then the signature will be rejected.

2.5.7. GOST Digital Signature

Another version of DSA algorithm is developed and used as standard by the Russian for message signing and verification. It also utilizes primes numbers p, q, y, s, etc as DSA with the following details (Schneier, B. 2000):

1. Prime number, p having length either between 509 to 512 or 1020 to1024 bits.
2. Prime factor q, such that its value less than p-1, namely in the range from 254 bits to 256-bits long.
3. α (which corresponds to g in NIST-DSA). It is an integer with value less than p-1, such that $a^q mod p = 1$.
4. An integer *x*, such that *x* < q. It is considered the private key for the signer.
5. The public key for the signer, y is calculated by the following equation.

$$y = a^x mod p, \tag{2.16}$$

The parameters p, q, g, and the public key y are all public, together with the hash function H.

2.5.7.1. Signing process

The signing process for GOST consists of calculating the signature parameters, r and s as follows.

$$r = (a^k mod p) mod q \text{ k} \tag{2.17}$$

$$s = (x.r + k\big(H(m)\big) mod q. \tag{2.18}$$

2.5.7.2 Verification process

The signature verification for GOST algorithm can be achieved by the following calculations

$$v = H(m)^{q-2} mod q \quad (2.19)$$

$$z_1 = (s.v) mod q. \quad (2.20)$$

$$z_2 = \big((q - r) * v\big) mod q. \quad (2.21)$$

Now, z1 and z2 are substituted in the following equation to produce u.

$$u = \big((a^{z1} * \ y^{z2}) mod p\big) mod q. \quad (2.22)$$

If u=r, then the signature is authentic and the message is accepted, otherwise it is rejected.

2.5.8. Yen-Laih Digital Signature

A DSA variant which attempt to create a faster signature by computing the inverse of the fixed private key x in advance and using it for each signature (Yen and Laih, 1995). The processes of the Signature and verification are illustrated as follows:

2.5.8.1 Yen-Laih Signature Generating Process

The message m is first hashed using the hash function H and signed by calculating r and s as in the following equations.

$$r = (g^k mod p)\ mod q \quad (2.23)$$

$$s = \Big(\big(r.k - h(m)\big).x^{-1}\Big)\ mod q. \quad (2.24)$$

The calculated signature r and s are then sent to the recipient, together with the message **m**.

2.5.8.2. Yen-Laih Verification Process

To verify the message signature, the verifier has to calculate u, using the following equations.

$$w = r^{-1} mod q. \quad (2.25)$$

$$u1 = (w.\big(h(m)\big) mod q. \quad (2.26)$$

$$u2 = (w.s) mod q. \quad (2.27)$$

Then substitute u1 and u2 from equations (2.26) and (2.27) into equation 2.28 in order to calculate *v*.

$$v = \big((g^{u1}.y^{u2}) mod p\big) mod q. \quad (2.28)$$

Now if v=r, signature is authentic, otherwise it is not accepted.

2.5.9. McCurley Digital Signature

Within this algorithm, the DSA verification process has been developed by eradicating the inverse from the computations on the verifier side to reduce the time complexity in order to verify the signature. The processes of the calculation of the signature and verification are as follows:

2.5.9.1. McCurley Signature Generating Process

To generate signature in this algorithm the signer has to calculate the following:

$$r = (g^k mod p)\ mod q. \quad (2.29)$$

$$s = (k.(h(m) + \ x\ .r^{-1}) mod q. \quad (2.30)$$

2.5.9.2. McCurley Verification Process

The verifier has to calculate u1 and u2 from equations 2.31 and 2.33, then compute *v* using equation 2.33.

$$u1 = (h(m) \,.s) mod q. \quad (2.31)$$

$$u2 = (s.r) mod q. \quad (2.32)$$

$$v = \big((g^{u1}.y^{u2}) mod p\big) mod q. \quad (2.33)$$

If v=r, then the message is authentic.

2.6. Review of Related Literature

To develop the signature processing time, several researchers have proposed some variant of DSA. Thus, some of these variant will be listed here:

1. **(Ali, 2004):** this researcher offered a developed version of DSA that reduced the processing time on the verification side, whilst the signing time stayed the same as that for NIST-DSA. This achievement was by minimizing the number of exponentiation in the used equations without any change to the parameter used in the NIST-DSA.
2. **(Poulakis, 2009):** suggested a modified variant of DSA which was based on a factorization problem and known as digital signature algorithm. It resists all the identified attacks, employs two discrete logarithms and maintains security strength at least equal to the original DSA. In order to present two modular exponentiations and a modular multiplication for the signature, this work has manipulated the properties of RSA technique.

3. **(Nguyen et al, 2011):** submitted a paper mainly on functionality Extension of the Digital Signature Standards. The utilized protocol here is on the ground of Belarusian digital signature standards due to its flexibility and the possibility providing natural extension of their functionality.
4. **(Galego Hernandes Jr, Carvalho and Proenca Jr, 2014):** employed Digital Signature to underpin network management for aiding network administrators through traffic characterization. Researchers have used the Digital Signature of network segment using flow analysis as a method for explaining standard network behavior; and genetic algorithm for optimizing the process.
5. **(Singh, Kaur and Kakkar, 2015):** presented a scheme for discovering any tampering and supporting the image compression. Researchers recommended that there is a transmission of a digital signature along with the image itself while the receiver will reproduce the signature in a correspondence to the received image. They also added that if both signatures are matching to each other, then the received image is authentic.
6. **(Alpizer-chacon and Chacon-Rivas, 2016):** applied digital signature to find a solution for the problem of verification of the authors and the contents of learn objects in online education via different repositories. They suggested that the author introduce his/her own digital certification by uploading the learn object to repository.
7. **(Dhagat and Joshi, 2016):** described a technique for digital signature in utilizing another singer as proxy. This scheme enables the sender to assign his signature to another signer, declaring that their scheme will offer protection to proxy signer private key. So, the proxy signer can put his/her signature instead of the original signer only within the validation period, as this scheme is controlled by certification holding identity of signer, giving

duration and imposing rules on the signing ability delegation by the original signer. Consequently, signer and proxy cannot deny each other as the scheme employs protected nominative signature.

8. **(SadrHaghighi and Khorsandi, 2016):** Utilized digital signature for detecting pollution attack in intra session Network coding and presented "Identity-based Digital Signature scheme", They asserted that in using this signature, the intermediate nodes can sense bogus packets and erase them before being combined with other packets. Additionally, the sender can keep up to date its own keys with no change of the identity and there is no necessity for a certificate management. Furthermore, the process of the verification is much faster in this scheme in a comparison with previous work.
9. **(M. Rifaat, 2017):** proposed a modified version of Digital Signature Algorithm (DSA which referred to as M.DSA) with a mathematical proof to improve the time complexity measurement. An investigation for the DSA and four of the variations has been utilized to scrutinize the impact of the digital signature parameters variations and their performance. The average time for singing and verification for the original and modified DSA and all other variants were computed for various parameter lengths of private keys, randomly generated keys and messages with a comparison of all the results. The findings of the study revealed that M.DSA has superior validation time in a comparison to others and the overall time complexity was impressive with speed gain about two minutes the original DSA overall time. The study has recommended that this modified version of DSA would be of used for applications which require fast verification time.

Chapter Three

Methodology and The Proposed Work

Chapter Three

Methodology and The Proposed Work

3.1. Introduction

This chapter is intended to outline the proposed modification scheme to the GOST digital signature algorithm. As the main purpose of the modification is to reduce the signature verification time, the procedure for determining the execution time for both signing and verification is outlined first, then the original GOST algorithm is investigated for various operational parameters, followed by investigation of the modified algorithm (M.GOST) using the same parameters. A mathematical proof of the correctness of the modified scheme is also included. Finally, few example are listed. These investigations have demonstrated an improvement in reducing the signature verification time.

3.2. The Methodology

GOST's principal design criterion doesn't seem to be computationally balanced, as the signing of a message is fast compared with DSA algorithm and its variants while the signature verification is much slower than others. It also has a key of double the size of that for DSA. This is mainly due to the usage of the modulus q which is at least 255-bit long. During verification, the modular inverses are computed by exponentiation and the generation of the public parameters is much more complicate than in case of DSA. This choice of the parameters makes GOST algorithm more secure as compared with DSA, obviously at the price of longer verification time.

The reason for fast signing process in GOST algorithm is that signers don't have to generate modular inverses as the basic signature equation for calculating signature parameter, s is:

$$s := xr + k.H(m) \bmod q \quad (3.1)$$

as compare with that for DSA, which is

$$s := (k^{-1}H(m) + xr) \bmod q \quad (3.2)$$

Also, the hash function for GOST algorithm is the Russian equivalent of the SHA.

This chapter includes two parts; the first part is an investigation of the GOST algorithm for message signing and message verification using various parameters and key lengths, while the second part present the new modified version of GOST that is suggested to improve the time measurement of the digital signature efficiency for signing and signature verification speed measurements.

3.3. Investigation of GOST algorithm

3.3.1. Initialization

During the initialization of the system, a trusted authority generates two primes *p,q* with the constraint that *q*| (*p*−1), and the public key generatorαis of order *q*. These values are kept the same for the whole session (for the sake of comparison). Hence, the proposed modified GOST digital signature algorithm, the original GOST algorithm, as well as any other digital signature algorithm considered here will be investigated using the same parameters: p, q, α, and the hash function H. However, the signer's private key (x), public key (y), and the random number (k) are selected and changed as required.

The investigation process will follow the flow chart shown in figure 3.1-a and figure 3.1-b in order to calculate the execution time for signing messages as well as their verification time of different message sizes and contents using a range of parameter values

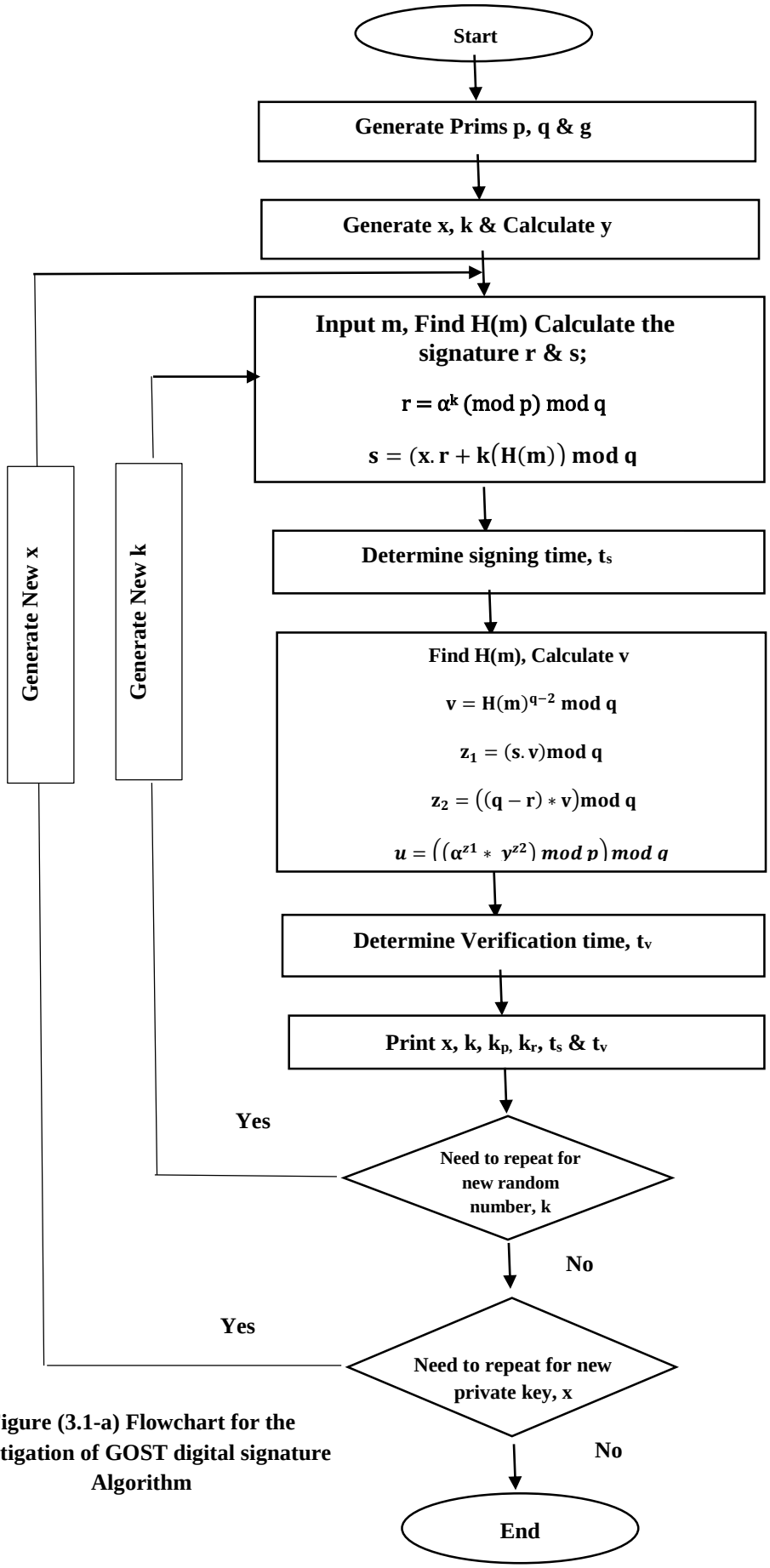


Figure (3.1-a) Flowchart for the investigation of GOST digital signature Algorithm

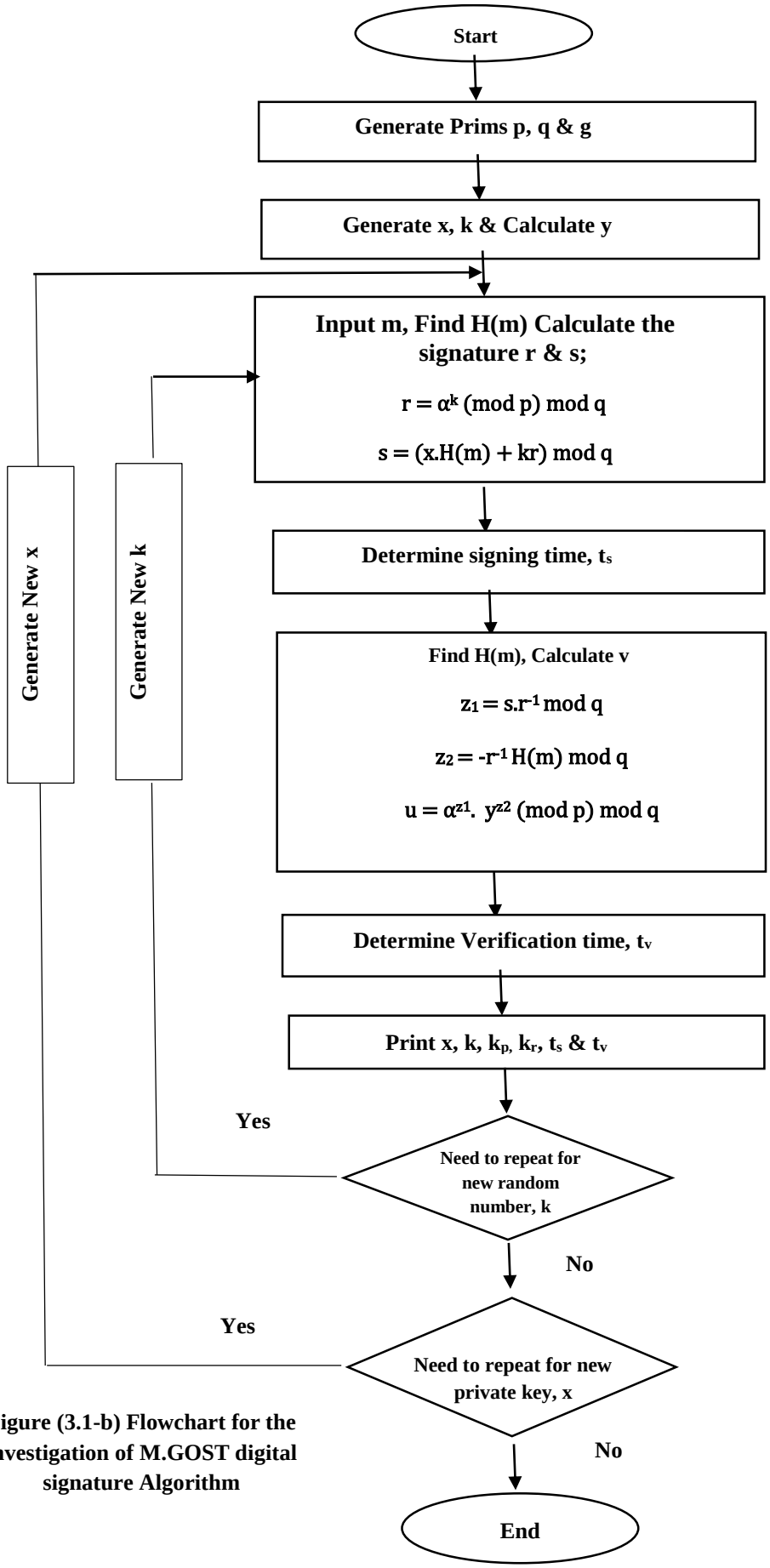


Figure (3.1-b) Flowchart for the investigation of M.GOST digital signature Algorithm

3.3.2. Public Key Generation

This procedure first determines the public key generator α after p and q have been determined.

Select a random number $d \in Z_p^*$

Calculate $f := d^{(p-1)/q} \bmod p$ (3.3)

If f=1, then go to step1.If $f \neq 1$thenα: =f.

Then, the public key for the signer is generated as follows:

A trusted authority chooses a large prime p, with $2^{509} < p < 2^{512}$or $2^{1020} < p < 2^{1024}$,and a prime divisor q, $2^{254} < q < 2^{256}$,with q| (p−1) .Every user chooses a secret key $x \in Z_q$ and computes his related public key by

$$y := \alpha^x \bmod p \quad (3.4)$$

To guarantee the security of the system, the public key so f the user must be certified by a trusted authority although this is not explicitly mentioned in the standard.

3.3.3. Signature generation

The signature generation for the message m is done by the following algorithm.

Calculate $H(m)$, the hash value of the message m, using the GOST hash function.

If $H(m) \equiv 0 \pmod q$ then set $H(m) := 0^{255-1}$

Create the random integer $k \in z_p^*$

Calculate the two values, $r' := \alpha^k \bmod p$

Then $$r := r' \bmod q = (\alpha^k \bmod p) \bmod q \quad (3.5)$$

Now r is one of the first elements of the signature, however if r= 0, pass to step 2 and create another random number k.

Using the signer's secret key x, calculate the second element of the signature s using equation 3.6.

$$s := (x.r + k.H(m)) \bmod q \quad (3.6)$$

If s=0, go to step 2.

The signature of the message m is the tuple (r and s) are then sent to the verifier.

3.3.4. Signature Verification

The verifier checks the authenticity of the message by checking the validity of the signature. This is possible, if he/she knows the public key of the signer. The signature verification consists of the following steps.

1. Verify the conditions $0<s<q$ and $0<r<q$. If one of the second conditions is not satisfied, the signature is not valid.
2. Calculate $H(m)$, the hash value of the received message m, but if $H(m) \equiv 0 \pmod q$, then set $H(m) := 0^{255-1}$
3. The verification equation is

$$r \equiv (\alpha^{s.H(m)}.y^{-r.H(m)}) \bmod q \quad (3.7)$$

It can be checked by the following steps:

1. Calculate the value v *as*

$$v = H(m)^{q-2} \bmod q \,. \quad (3.8)$$

Which is the multiplicative inverse of $H(m) \bmod q$.

2. Compute the values $z_1 := s.v \bmod q$ (3.9)

And $z_2 := x^{q-r}.v \bmod q$ (3.10)

3. Calculate the value

$$u := (\alpha^{z1}.y^{z2} \bmod p) \bmod q \quad (3.11)$$

4. Verify the condition $r=u$. If this check succeeds, the message is accepted as authentic, i.e. it is really signed by the claimed signer, and otherwise it is rejected.

3.3.5. Signature Validation Proof

In order to have $r=u$, z_1+xz_2 should equal to k, therefore, substitution for z_1 and $z2$ gives

$$s.v \bmod q + x.x^{q-r}.v \bmod q$$

$$= (x.r + k.H(m).H^{-1}(m) + x.H^{-1}(m).(q-r)) \bmod q \quad (3.12)$$

But $(q-r) \bmod q = -r \bmod q$.

Substitution of the last equation results into:

$$= x.r.H^{-1}(m) + k.H(m).H^{-1}(m).-r \bmod q = k \quad (3.13)$$

(q.e.d.)

3.4. Proposed Measurements

- Security tests will be conducted to be satisfied with the modified scheme.
- Comprehensive comparison of the suggested scheme will be conducted with DSA and its variants.

3.5. Modification of GOST

The modified M.GOST algorithm is aimed to produce improvements in the signature verification processing time. Few alterations to the equations used for the calculation is done that reduces the execution computation steps without affecting the security parameters that were used for the original GOST algorithm. These alterations are written in details in the following sections that include signing and verification. The mathematical proof of the modified verification process is included next followed by a numerical example.

3.5.1. Create the digital Signature

Let the signature be as follows:

Having the message m,

1- Calculate the hash value of m, using SHA algorithm, i.e. h=H(m). (3.14)

2- Generate a random integer k for each message such that k < q.

3- Calculate the signature r and s as follows:

$$r = (\alpha^k \bmod p) \bmod q \quad (3.15)$$

$$s = (x.H(m) + k.r) \bmod q \,. \quad (3.16)$$

4- The digital signature r & s parameters are sent to the verifier, together with message m.

3.5.2. Validate the signature

1- On receiving r, s, & m, calculate H(m), then determine $H^{-1}(m)$.

2- Let $z_1 := s.r^{-1} \bmod q$ and . (3.17)

$$z_2 := -H(m).r^{-1}.\bmod q \,. \quad (3.18)$$

3- Now calculate $u = (\alpha^{z1}.y^{z2} \bmod p) \bmod q$). (3. 19)

If **u=r**, then the signature is accepted, otherwise it is rejected.

3.5.3. Mathematical Proof

The received signature elements are ***r*** and ***s*** together with the message m. A mathematical expression ***u*** will be obtained that is to be compared with expression for ***r***, then exponents are matched, as will be seen in the following steps.

1- Let $u = (\alpha^{z1}.y^{z2} \bmod p) \bmod q$ (3.20)

$= (\alpha^{z1}.\alpha^{x.z2} \bmod p) \bmod q$ (3.21)

$= (\alpha^{z1+x.z2} \bmod p) \bmod q$ (3.22)

Substituting for z_1 and z_2, resulting into

$$u = (\alpha^{(s.r^{-1} - x.H(m).r^{-1})} \bmod p) \bmod q \quad (3.23)$$

but $r = \alpha^k \pmod p \bmod q$ (3.24)

Therefore, if $s.r^{-1} - x.H(m).r^{-1}$ equals to *k*, then **u=r** (3.25)

Substitute for *s* into the exponent of α in equation 3.23, the exponent becomes:

$$(x.H(m) + k.r).r^{-1} - x.H(m).r^{-1} \quad (3.26)$$

Now, simplifying equation 3.26 results into ***k***, i.e. **u = r**.

(q.e.d.)

3.6 Example of application of the GOST algorithm

1. Initialization

- p = 3023
- q = 1511.
- h= 1351

g*: Public key generator.

- $g = h^{p-1/q}$

  = 2332.

2. Key generation

- *x*=250.
- $y = g^x \bmod p$

  = $2332^{201} \bmod 3023$

  = 1991

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

3. Signature generation

Now we send the digitally signed message as follows

- *k* =2172.
- H(m) = 1702
- $r = g^k \bmod p$

  = $2332^{2172} \bmod 3023$

  = 451

  * Parameter (g) in DSA algorithm is the same parameter (α) in GOST algorithm

- $s = r + k, H(m).x^{-1}$

  = 451+2172(1702*250^-1)

  = 263

The signature for M is (r, s) = (451, 263).

4. Signature verification

Now we verify the Signature of sender (r, s) on message M as follows

- $u_1 = H(m).s'$

  = 1457

- $u_2 = H(m).r'$

  = 1121

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

  = $(1991^{1457-1121} \bmod 3023) \bmod 1511$

  = 451

Accept, where $v = r = 451$.

3.7 Example of application of the M.GOST algorithm

1. Initialization

- p = 3023
- q = 1511.
- h= 1351

g*: Public key generator.

- $g = h^{(p-1)/q}$

  = 2332.

2. Key generation

- *x*=387
- $y = g^x \bmod p$

  = $2332^{387} \bmod 3023$

  = 2034

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

3. Signature generation

Now we send the digitally signed message as follows

- *k* =156.
- H(m) = 505
- $r = g^k \bmod p$

  = $2332^{156} \bmod 3023$

  = 483

  * Parameter (g) in DSA algorithm is the same parameter (α) in GOST algorithm

- $s = r + k.H(m).x^{-1}$

  $= 483 + 156.505.387^{-1}$

  = 314

  The signature for M is (r, s) = (483, 314).

4. Signature verification

Now we verify the Signature of sender (r, s) on message M as follows

- $z_1 = r.s^{-1} \bmod q$

  = 326

- $z_2 = -H(m).r^{-1} \bmod q$

  = 390

- $v = (\alpha^{z_1}.y^{z_2} \bmod p) \bmod q$

  = 483

  Accept, where $v = r$**=483.**

Chapter Four

Implementation and Results

Chapter Four
Implementation and Results

4.1 Introduction

This chapter displays the implementation of GOST Algorithm and the proposed M.GOST algorithm, as well as a comparison with another four algorithm variants, namely Yen-Laih, McCurley, NIST-DSA, and M.DSA, which were examined, too. This examination included the signing and verification execution time for different keylengths and different message lengths and contents. For the coding and testing purpose, Microsoft Visual Studio Integrated Development Environment using C# language has been utilized in this study. The BigO values for all the considered algorithms were evaluated and they are compared with each other. Finally, the execution speed gains for the M.GOST over other algorithms are calculated and compare.

4.2 Digital Signature Implementation

This section tackles the digital signature implementation for both the standard GOST algorithm and the modified M.GOST algorithm in order to scrutinize and compare the suggested improved scheme with the standard GOST. This scrutinizing and comparison will be according to various values for the private key of the signer and the random integer number k that is used for each message. This study is carried out first for values of the randomly chosen prime numbers p, and divisor q, then integer values h to generate the values of g, which used to generate the signee public y as illustrated below in table 4.1.

Table (4.1) Parameters of GOST and M.GOST Algorithms with Prime values

Parameters	GOST	M.GOST
p	√	√
q	√	√
h	√	√
g	$g = h^{p-1/q}$	

From the table above, we can say that the GOST algorithm and M.GOST algorithm take the same parameters In this thesis, the values of p and q were selected with random primes values and h values were added to generate g values. These values of p, q, h were allocated the same values in GOST and M.GOST algorithms in order to be able to compare them correctly.

In above Table GOST and M.GOST algorithms take the same parameters (p, q and h values) to generate the value of g but these parameters take randomly primes values.

The signing time and the signature verification time are calculated for different values of the private key x in order to find different values of the corresponding public key.

For each value of the private key x, the signing time is measured for different values of the random integer number k. Then the average time is calculated for 10 different values of the random integer number k is calculated for different values of the random integer number k. The signature verification process is done for each of the signed messages and their execution time is measured too.

This test is done to see the effect of key length on the time measurement. The obtained results are listed in table 4.2

Table (4.2) comparison of execution time for different key lengths for GOST signature and verification

No. of digits	x Private Key	k Random number for each message	Execution time in (msec)	
			Signing	Signature Verification
2 Digits	36	36	0.44	0.89
		142	0.35	0.89
		153	0.33	0.93
		390	0.44	11.1
		721	0.35	10.2
		858	0.44	0.93
		1366	0.44	0.98
		1749	0.44	10.2
		2223	0.44	0.98
		2571	0.44	10.2

3 Digits	772	250	0.35	10.2
		434	0.35	0.80
		731	0.40	10.2
		1112	0.40	0.93
		1345	0.44	0.98
		1572	0.40	10.2
		2065	0.40	0.80
		2130	0.44	11.1
		2248	0.53	11.1
		2517	0.44	10.7
4 Digits	1017	84	0.31	0.98
		1066	0.40	10.2
		1962	0.40	0.98
		2197	0.44	0.93
		2297	0.40	0.93
		2787	0.44	0.93
		2799	0.44	10.2
		2867	0.40	0.93
		2917	0.44	10.2
		2927	0.49	0.98
Average Time			0.41	5.05

Table (4.3) comparison of execution time for different key lengths for M.GOST signature and verification

No. of digits	x Private Key	k Random number for each message	Execution time in (msec)	
			Signing	Signature Verification
2 Digits	71	302	0.22	0.35
		544	0.17	0.35
		902	0.22	0.35
		1121	0.22	0.35
		1532	0.26	0.40
		1946	0.26	0.40
		2189	0.22	0.44
		2265	0.26	0.35
		2510	0.26	0.40
		2613	0.26	0.40

3 Digits	244	42	0.26	0.75
		219	0.35	0.66
		355	0.35	0.66
		690	0.35	0.62
		743	0.53	0.71
		1134	0.44	0.53
		1175	0.44	0.62
		1305	0.40	0.66
		1413	0.40	0.62
		2059	0.40	0.66

4 Digits	1115	2210	0.53	0.71
		491	0.44	0.53
		975	0.58	0.71
		1045	0.44	0.84
		1136	0.94	0.71
		1873	0.58	0.75
		2185	0.53	0.80
		2397	0.53	0.66
		2797	0.58	0.84
		2902	0.49	0.80
Average Time			0.40	0.59

Different message lengths are used for testing the execution time for both the original (GOST) algorithm and modified algorithm (M.GOST) and the results are listed in table (4.4) and table (4.5). Message lengths used were (20, 40, 60, 80 and 100 characters) and the private key length of 100 digits.

Table (4.4) Signing Time Calculation for GOST & M.GOST signature with p, q and g of Length of 100 Digits

Length of Message Characters	The used message *	GOST Average Signature time in (msec)	M.GOST Average Signature time in (msec)
20	I am extremely happy	37.28	13.93
40	The weather was beautiful yesterday	32.05	36.91
60	English language is one of the international languages today	29.33	41.30
80	Some people preferably choose to use the private vehicles due to the convenience	28.47	42.18
100	The governments should encourage their residents to use public transportations as it brings benefits	28.19	23.74
Overall average time		31.06	31.61

* The space is considered as character

The figure (4.1) below illustrates the accuracy of the results.

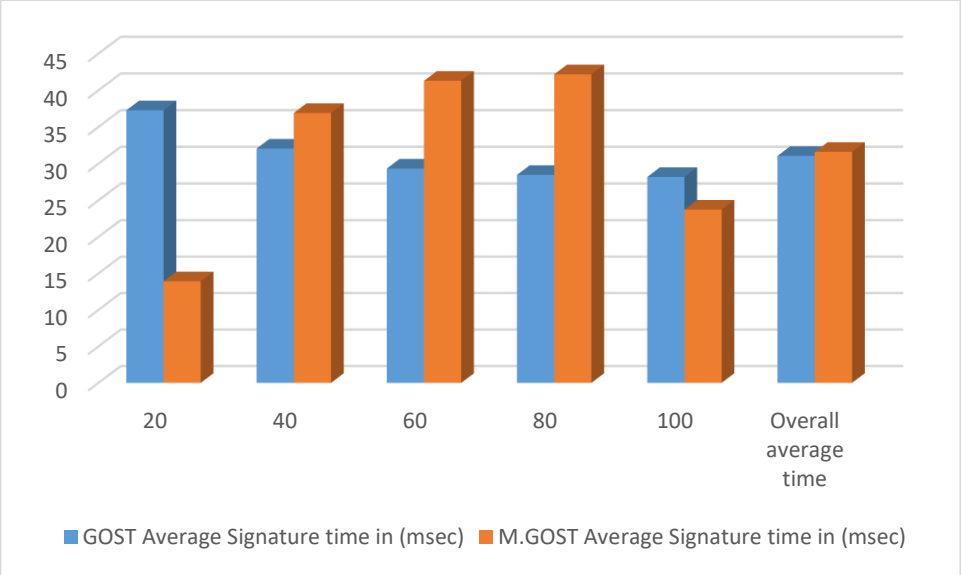


Figure (4.1) Signing Time Calculation for GOST & M.GOST signature with p, q and g of Length of 100 Digits

Table (4.5) Verification Time Calculation for GOST & M.GOST verification with p, q and g of Length of 100 Digits

Length of Message characters	The used message *	GOST Average Verification time in (msec)	M.GOST Average Verification time (msec)
20	I am extremely happy	86.97	22.66
40	The weather was beautiful yesterday	74.43	46.36
60	English language is one of the international languages today	67.62	64.34
80	Some people preferably choose to use the private vehicles due to the convenience	65.47	65.67
100	The governments should encourage their residents to use public transportations as it brings benefits	64.55	37.70
Overall average time		71.81	47.34

* The space is considered as character

The figure (4.2) below illustrates the accuracy of the results.

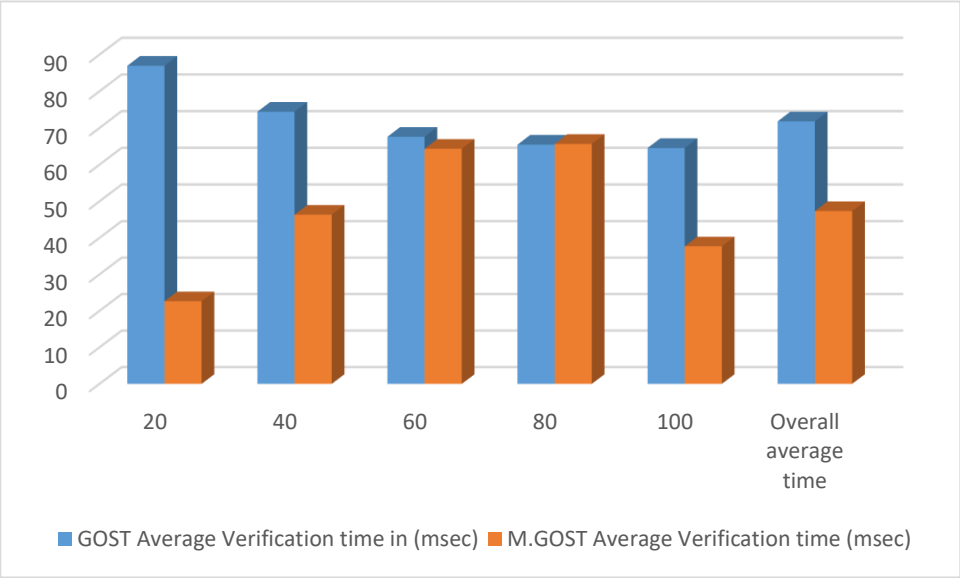


Figure (4.2) Verification Time Calculation for GOST & M.GOST verification with p, q and g of Length of 100 Digits

The execution time for different message lengths does note very with message length, because only the hash values of the message is involved in the calculations which is always of the same length. However, the noticed variation in the time can be attributed to the computation environment time complexity. Usually the average time is compered rather than the time itself in order to have more realistic comparison.

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 100 different keys) in this thesis, the algorithms that are considered for comparison with the proposed algorithm in this thesis (i.e. M.GOST) to be compared with are: GOST (Bruce, S. ,1996), DSA (NIST), M.DSA (M. Rifaat, 2017**)**, Yen-Laih (Yen, S. M., & Laih, 1995), and McCurley (Schnier,1996).

The results summary of the calculations is listed in table (4.6). While the detailed calculation results for each algorithm.

Table (4.6) The Average of signing and verification Times in Comparison for M.GOST with other DSA algorithms

NO.	Algorithms	Avg. signature time (msec)	Avg. verification time (msec)
1.	GOST	31.06	71.81
2.	M.GOST	31.61	47.34
3.	McCurley	36.05	47.66
4.	Yen-Laih	36.00	51.22
5.	DSA	35.86	50.88
6.	M.DSA	36.52	23.86

The figure (4.3) below illustrates the accuracy of the results.

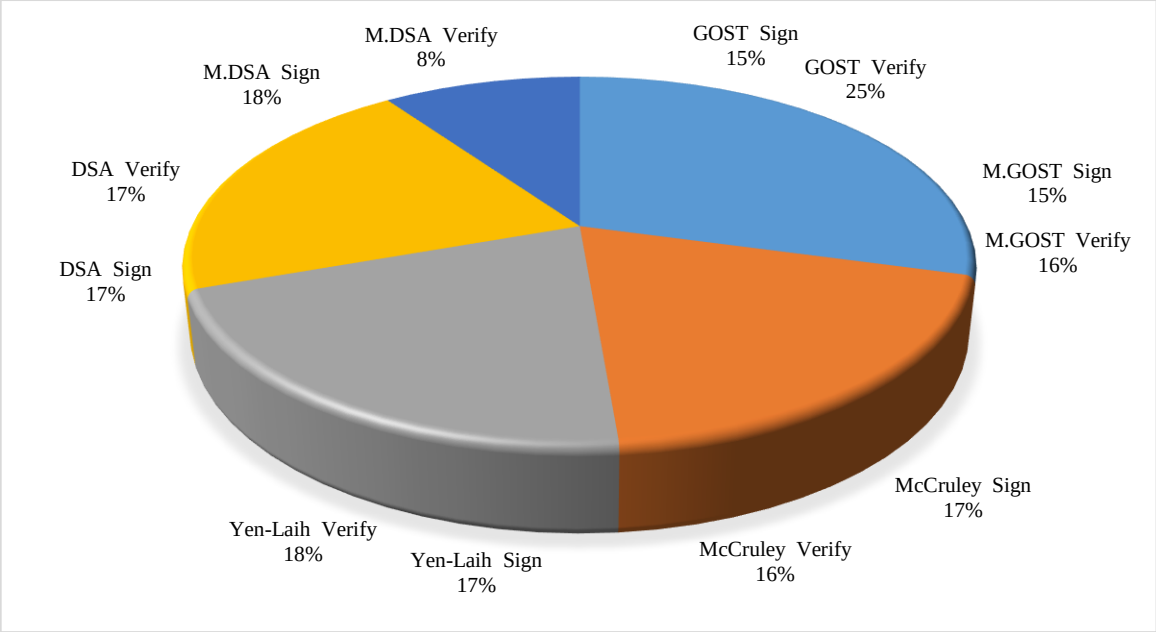


Figure (4.3) The Average signing and verification Times Comparison for Several Algorithms with M.GOST

In order to visualize the execution time clearly, the obtained values of Table 4.6 are normalized to the highest execution time and listed in Table 4.7, the plotted in Figure 4.4 for both signing and verification processes. It shows the improvement in the verification time for M.GOST algorithm compared with that for GOST and how it compares with other DSA algorithms. I must be noted that these measurements were taken for signing and verification of the same message using the same parameters on the same computing environment.

Table (4.7). The Normalized signing and verification execution time comparison for M.GOST and other algorithms (Normalized to the highest execution time)

NO.	Algorithms	Signing time (msec)	Verification time (msec)
1.	GOST	0.433	1
2.	M.GOST	0.440	0.659
3.	McCurley	0.502	0.664
4.	Yen-Laih	0.501	0.713
5.	DSA	0.499	0.709
6.	M.DSA	0.509	0.332

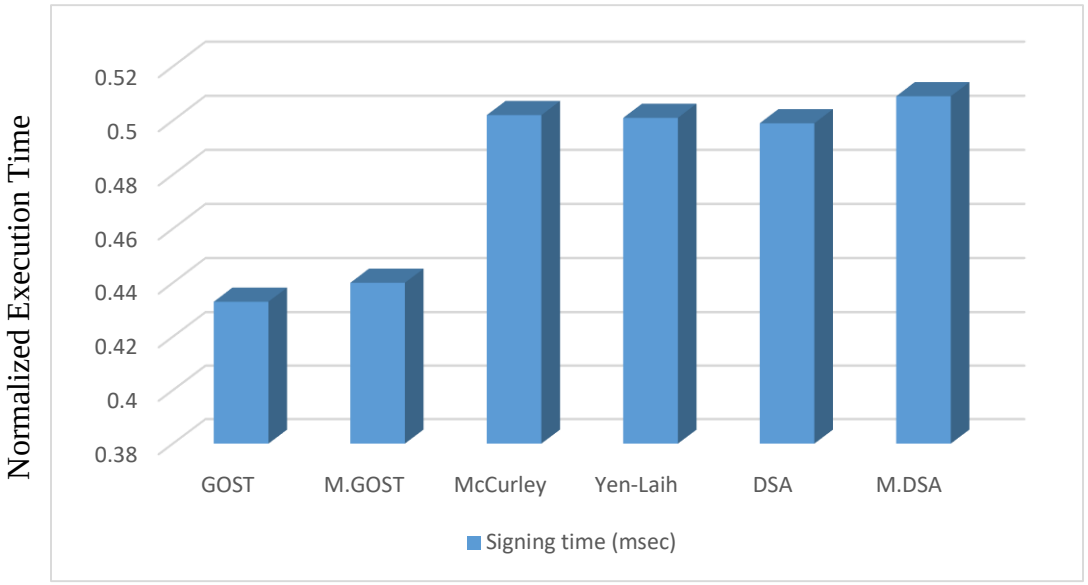


Figure (4.4-a) Comparison of the normalized signing time for M.GOST with other algorithms.

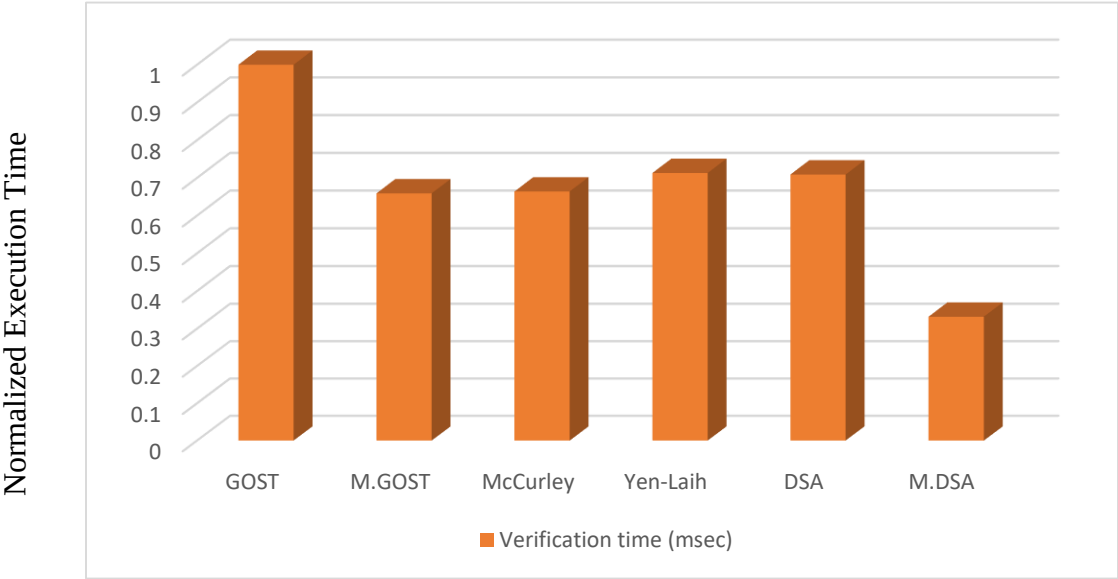


Figure (4.4-b) Comparison of the normalized verification time for M.GOST with other algorithms.

4.4. Algorithms Signature and Verify complexity (M. Rifaat, 2017)

One of the most significant measures for cryptographic algorithm development is the time complexity where Big **O** notation is the most well-known usage measure. It categorizes the cryptographic algorithm into the principle classes of algorithm time complexity. This section will present the GOST signature of the message hash that includes two values r and s as a measure of each Digital Signature Algorithm. A complexity calculation will be functioned for both of them for GOST, M.GOST, DSA, M.DSA, McCurley, and Yen-Laih algorithms. Later, a comprehensive complexity of the signature will be done for all of algorithms signature complexity.

4.4.1. GOST Signature complexity

$$r = (\alpha^k \bmod p) \bmod q \quad (4.1)$$

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

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

$$s = (x.r + k.H(m)) \bmod q \quad (4.2)$$

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

$(s) = 1.$

Therefore,

$(siganture) = (\log n)$

4.4.2. GOST Verify complexity

GOST verification has four steps as follows:

$$v = H(m)^{q-2} \bmod q \quad (4.3)$$

$$z_1 = (s.v) \bmod q \quad (4.4)$$

$$z_2 = (q - r).v \bmod q \quad (4.5)$$

$$u = (\alpha^{z_1}.y^{z_2} \bmod p) \bmod q \quad (4.6)$$

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

$O(v) = (logn)$

$O(z_1) = O(s.v \bmod q)$

$(z1) = 1.$

$O(z_2) = O((q-r).v \bmod q)$

$(z2) = 1.$

$O(u) = O((\alpha^{z_1}.y^{z_2}) \bmod p) \bmod q$

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

Therefore,

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

$(verify) = (3\log n)$

4.4.3. M.GOST Signature complexity

$r = (\alpha^k \bmod p) \bmod q$ (4.7)

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

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

$s = (x.h + k.r) \bmod q$ (4.8)

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

$(s) = 1$.

Therefore,

$(siganture) = (\log n)$

4.4.4. M.GOST Verify complexity

M.GOST verification has three steps as follows:

$$z_1 = s.r^{-1} \bmod q \tag{4.9}$$

$$O(z_1) = O(s.r^{-1} \bmod q)$$

$$O(z_1) = 1$$

$$z_2 = (-r^{-1}.h) \bmod q \tag{4.10}$$

$$O(z_2) = O((-r^{-1}.h) \bmod q)$$

$$O(z_2) = 1.$$

$$u = (\alpha^{z_1}.y^{z_2} \bmod p) \bmod q \tag{4.11}$$

$$O(u) = O((\alpha^{z_1}.y^{z_2} \bmod p) \bmod q)$$

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

Therefore,

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

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

Table (4.8) Summary of the result of computations previous algorithm and compare their Big O complexity

NO.	Algorithm	Avg. signature time (µsec)	Signature Big **O**	Avg. verification time (µsec)	Verification Big **O**	Total Big **O**
1.	GOST	31.06	Logn	71.81	3logn	4logn
2.	M.GOST	31.61	Logn	47.34	2logn	3logn
3.	McCurley*	36.05	2logn	47.66	2logn	4logn
4.	Yen-Laih*	36.00	2logn	51.22	3logn	5logn
5.	DSA*	35.86	2logn	50.88	3logn	5logn
6.	M.DSA*	36.52	2logn	23.86	Logn	3logn

* The results of the (McCurley, Yen-Laih, DSA, M.DSA) algorithms were obtained from (M. Rifaat, 2017**)**

4.5. Signature and Verification Speed of GOST and M.GOST Algorithms

The execution speed improvement in the signature and / or the verification processes for the M.GOST will be calculated and compared to those for the other algorithms, by dividing the average of GOST algorithm execution time by the average time of M.GOST. As indicated in the following tables.

4.5.1. Signature Speed Gain for M.GOST to other algorithms

The attempts to modify GOST's signing execution time is to have the best speed. Success in speeding up is one goal, but nevertheless can't achieve gains for all goals such as making improvements for both signing and signature validation. The Signing Speed Gain (*SSG*) of M.GOST over GOST algorithm for example can be computed by equation 4.12.

$$SSG = \frac{GOST-time}{M.GOST-time} * 100\% \qquad (4.12)$$

$$= \frac{31.06}{31.61} * 100\% = 98\%$$

In the same way, the signature speed for all other algorithms are calculated and listed in

Table (4.9) below

Table (4.9) Signature Speed Gain for M.GOST to other algorithms

NO.	Algorithm	Avg. signature time (msec)	M.GOST Avg. signature time (msec)	M.GOST Signature speed gain over other algorithm
1.	GOST	31.06	31.61	0.98
2.	McCurley	36.05	31.61	1.14
3.	Yen-Laih	36.00	31.61	1.14
4.	DSA	35.86	31.61	1.13
5.	M.DSA	36.52	31.61	1.16

If the obtained signing speed gain value is more than 1, it means that there is improvement in the execution speed. Hence less than 1 values mean there is deterioration in the execution speed.

4.5.2. Verification Speed Gain for M.GOST to other algorithms

M.GOST provides greater efficiency over GOST algorithm for the evaluated time of verification and also over some other algorithms. Verification Speed Gain (VSG) calculations for M.GOST algorithm with respect to GOST algorithm are achieved by equation 4.13.

$$VSG = \frac{GOST-time}{M.GOST-time} * 100\% \qquad (4.13)$$

$$= \frac{71.81}{47.34} * 100\% = 1.52$$

By the same way, the speed for verification is calculated for all other algorithms. The results are summarized in Table (4.10)

Table (4.10) Verification Speed Gain for M.GOST to other algorithms

NO.	Algorithm	Avg. verification time (msec)	M.GOST Avg. verification time (msec)	M.GOST speed over other algorithm
1.	GOST	71.81	47.34	1.52
2.	McCurley	47.66	47.34	1.01
3.	Yen-Laih	51.22	47.34	1.08
4.	DSA	50.88	47.34	1.07
5.	M.DSA	23.86	47.34	0.50

Chapter Five

Conclusions and Future Work

Chapter Five

Conclusions and Future Work

5.1. Conclusions

The purpose of this thesis is to improve the digital signature algorithm by reducing the complexity of signature time and the signature verification time. The authentication of the sender identity has been achieved by reducing the calculation steps in the original GOST while maintaining the same parameters strength. From practical experience, the following conclusions can be drawn:

- The digital signature mechanism provides authentication of the sender's identity in terms of the integrity of the data transmitted by the other party, even if the communication channel is unsafe.
- The proposed M.GOST model, by reducing the processing steps, showed a shorter verification time compared to the original GOST algorithm in terms of signature time and verification time as well as other algorithms such as DSA and its variants; Yen-Laih, and McCurley.
- The study showed that the time of signing for the M.GOST algorithm was faster, but the verification time was slower than the M.DSA algorithm.
- Big **O** accounts are included in the search to visualize the obtained optimization, which means an improvement in the time complexity of the algorithm

- It was observed that the time of the calculation fluctuates according to changes of the private key length and the generated random number, thus the average time for a large number of experiments is taken for all measurements. This indicates that the length of the private key, the random integer, and message does not affect the speed of the signature, but this action is the reason for the improvement in account time**.**

5.2. Future work

Future work on the digital signature may extend to the following problems:

- Implement the proposed GOST algorithm in certain applications such as commercial applications, e-government, e-banking and e-elections. Security and military.
- The implementation of this technique helps to control the systems of personal identification, intrusion detection and penetration of transmitted data.
- It may also be useful to improve them in coordination with biometrics to generate a signature.
- Due to the quick signature process of the GOST Digital Signature algorithm and the M.GOST Digital Signature algorithm and the application obtained in the signature validation process, these two algorithms can be combined to look for further improvements in the GOST that can be achieved for signing and verification.

References

- 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.
- 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 IEEE.
- Atreya Mohan, et al (2002). Digital Signatures. USA.
- 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 IEEE.
- Dhagat, R., & Joshi, P. (2016). New approach of user authentication using digital signature. In Colossal Data Analysis and Networking (CDAN), IEEE.
- Dhillon, Alan (2002). A Web-Based Tutorial on Digital.
- Diffie, W., & Hellman, M. (1976). New directions in cryptography. IEEE transactions on Information Theory, 22(6).
- 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 Springer Berlin Heidelberg.
- ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. In Workshop on the Theory and Application of Cryptographic Techniques. Springer Berlin Heidelberg.

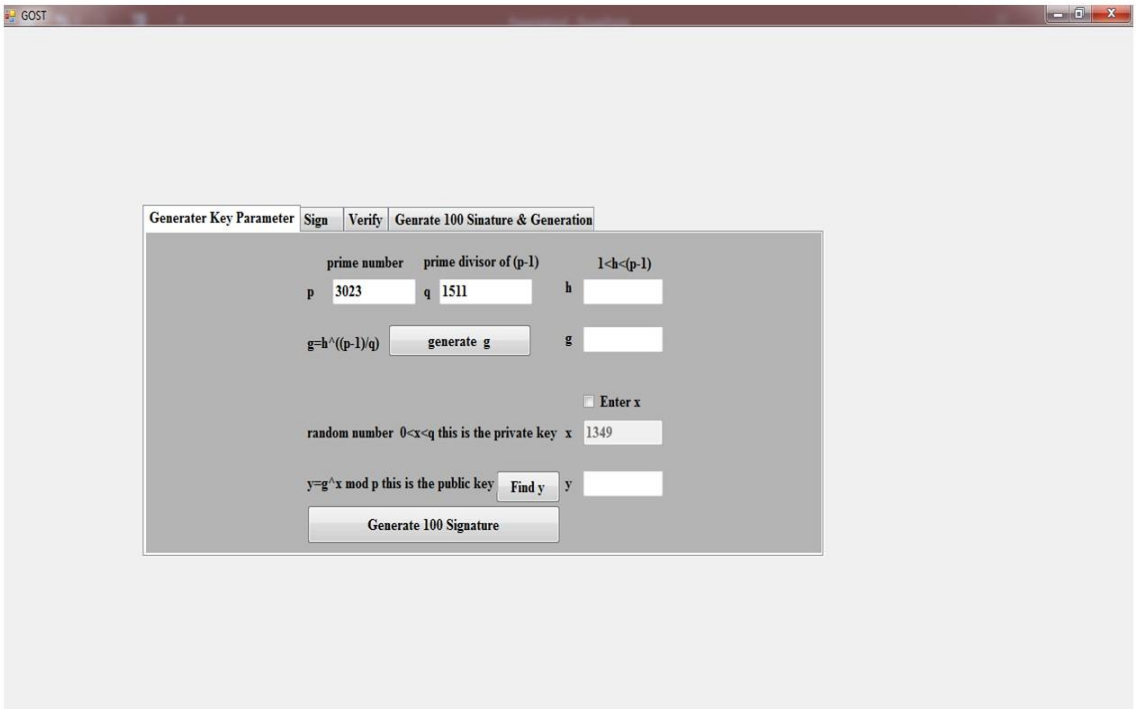
- Gallagher, P., & Kerry, C. Fips pub 186-4: Digital signature standard, DSS (2013).
- Hernandes, P. R. G., & Carvalho, L. F. (2014, November). Digital Signature of Network Segment Using Flow Analysis through Genetic Algorithm and ACO Metaheuristics. In Chilean Computer Science Society (SCCC), 2014 33rd International Conference of the. IEEE.
- IEEE Potentials, Volume: 25, Issue: 2, March-April 2006.
- Kerry, Cameron F. & Gallagher, Patrick D. (2013), Federal Information Processing Standers Publication, Digital Signature Standard (DSS), National Institute of Standards and Technology, Issue July 2013.
- Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1996). Handbook of applied cryptography. CRC press.
- Nguyen, Minh Hoai, et al, 2016, Deep learning for analyzing synchrotron data streams, New York Scientific Data Summit (NYSDS),IEEE Conferences
- Paul, Eliza (2017). Introduction to Digital Signatures.
- Poulakis, D. (2009). A variant of digital signature algorithm. Designs, codes and cryptography, 51(1).
- Merkle, R. C. (1989), A Certified Digital Signature, Advance in Cryptology-Crypto'89, Springer-Verlag'.
- Rifaat, Mohammed Mustafa (2017). Computation Complexity Improvement for Digital Signature Algorithm. (Unpublished Master's Thesis), Middle East University, Amman, Jordan.

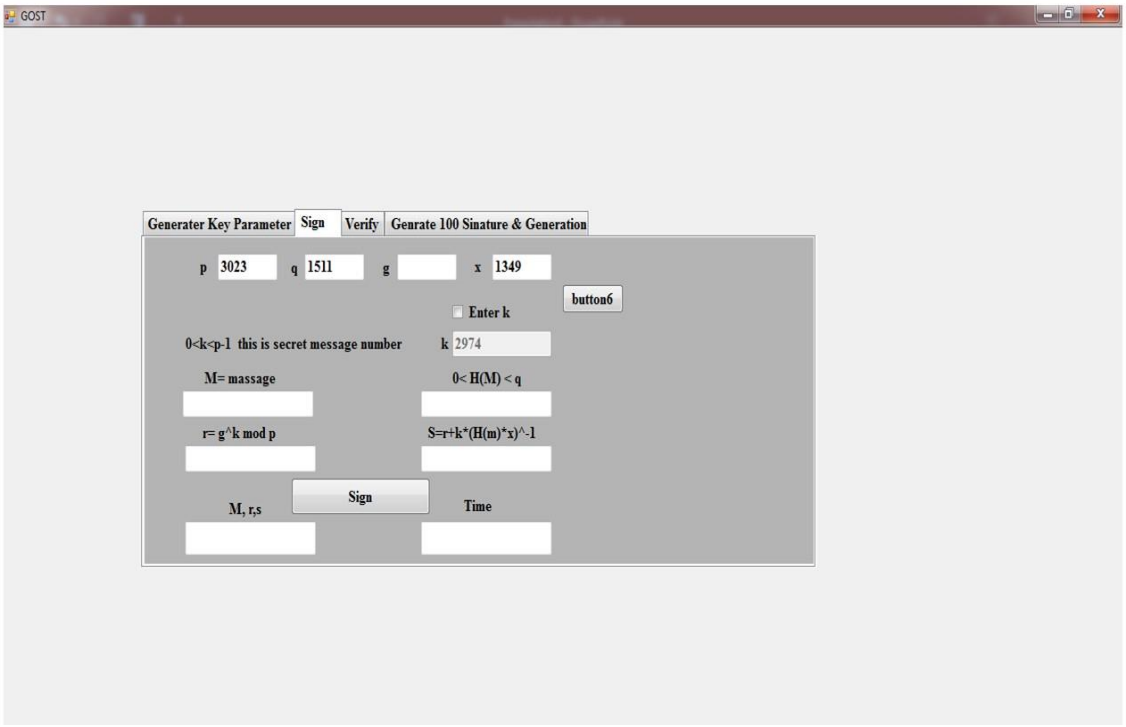
- Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2)
- S. Yen, C. Laih, Improved Digital Signature Suitable for Batch Verification, IEEE Transaction on Computers, vol.44, No.7, July 1995.
- SadrHaghighi, 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 IEEE.
- Schneier, B. (2000). Applied Cryptography Second Edition: protocols, algorithms, and source. Beijing: China MachinePress.
- 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.
- 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). IEEE.
- Stallings, W. (2006). Cryptography and Network Security: Principles and Practice.
- Xin, Li (2007), An Improvement of Diffie-Hellman Protocol, Network & Computer Security, vol. 12.

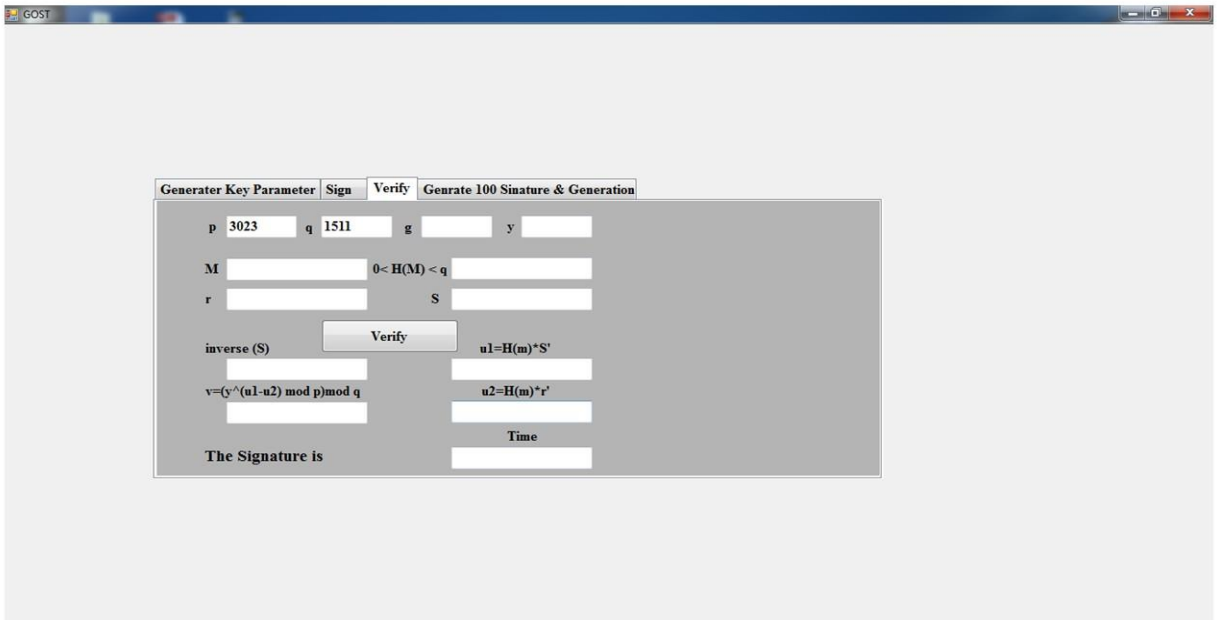
Appendix A

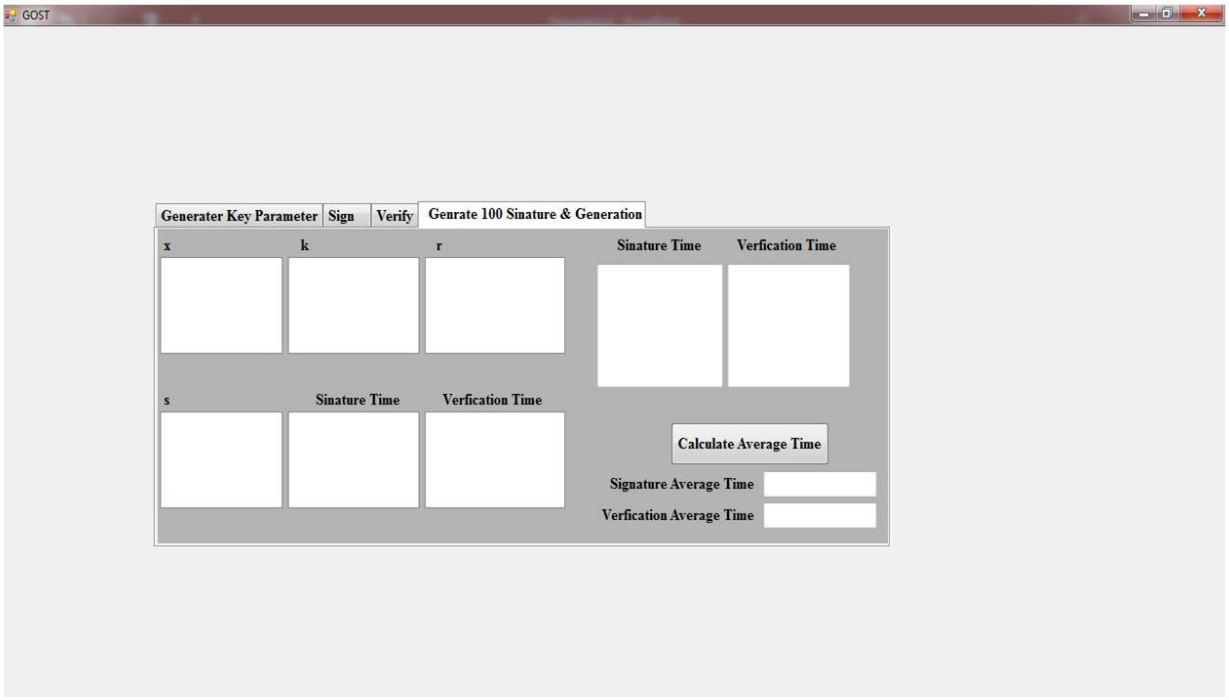
GOST & M.GOST Algorithms Interfaces

1- GOST Algorithm

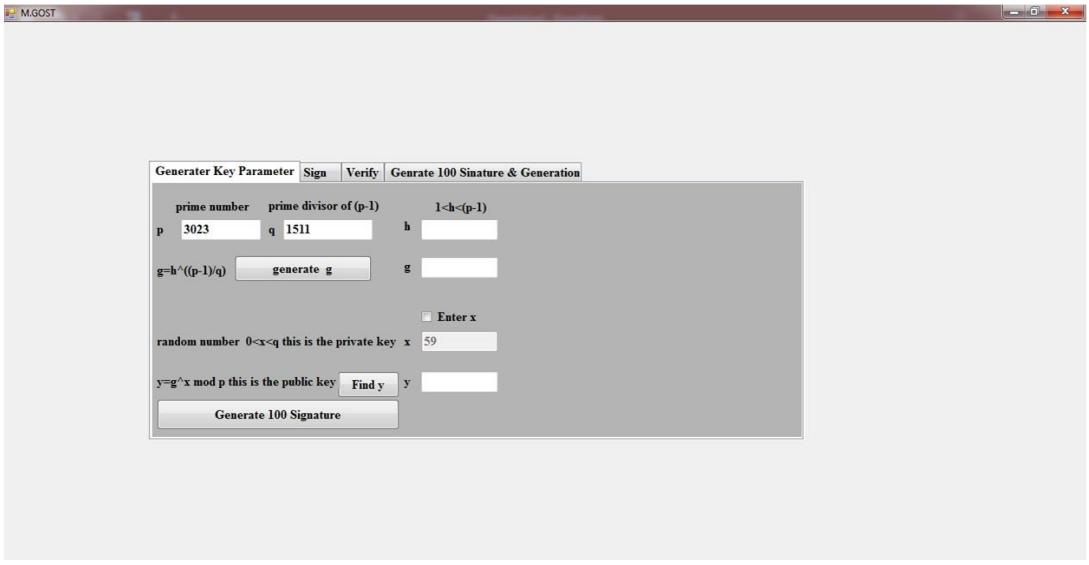


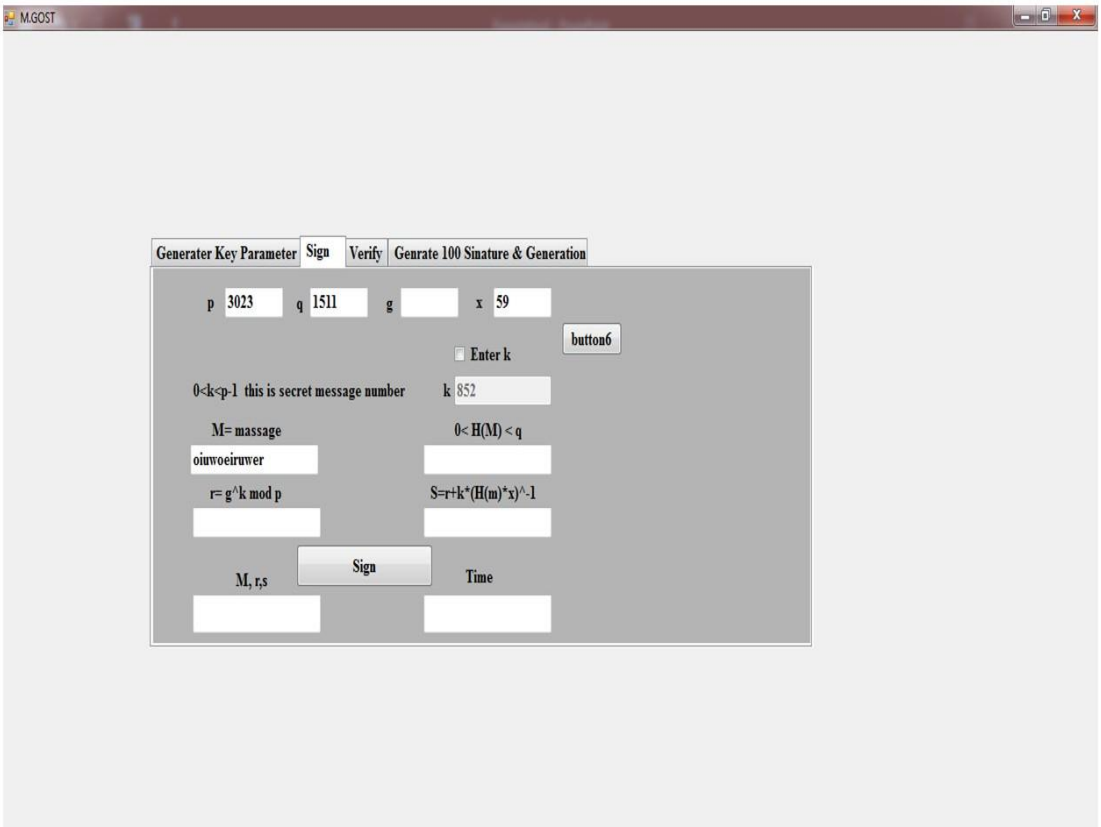


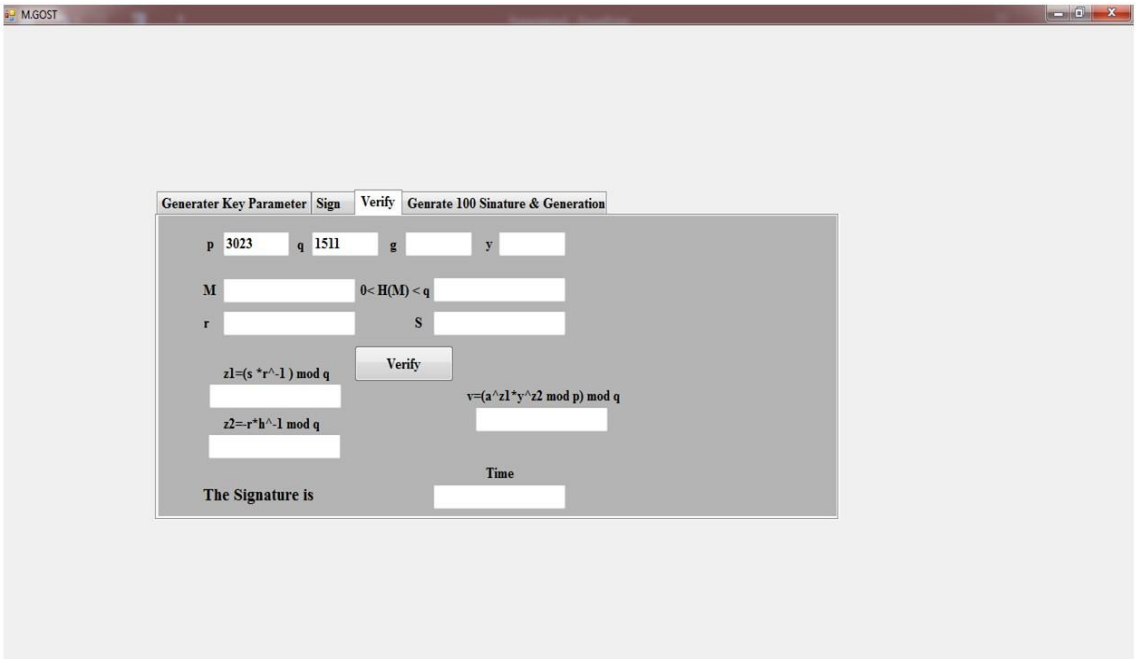

GOST
Generater Key Parameter
Sign
Verify
Genrate 100 Sinature & Generation
p
3023
q
1511
g
y
M
0< H(M) < q
r
S
Verify
inverse (S)
u1=H(m)*S'
v=(y^(u1-u2) mod p)mod q
u2=H(m)*r'
Time
The Signature is

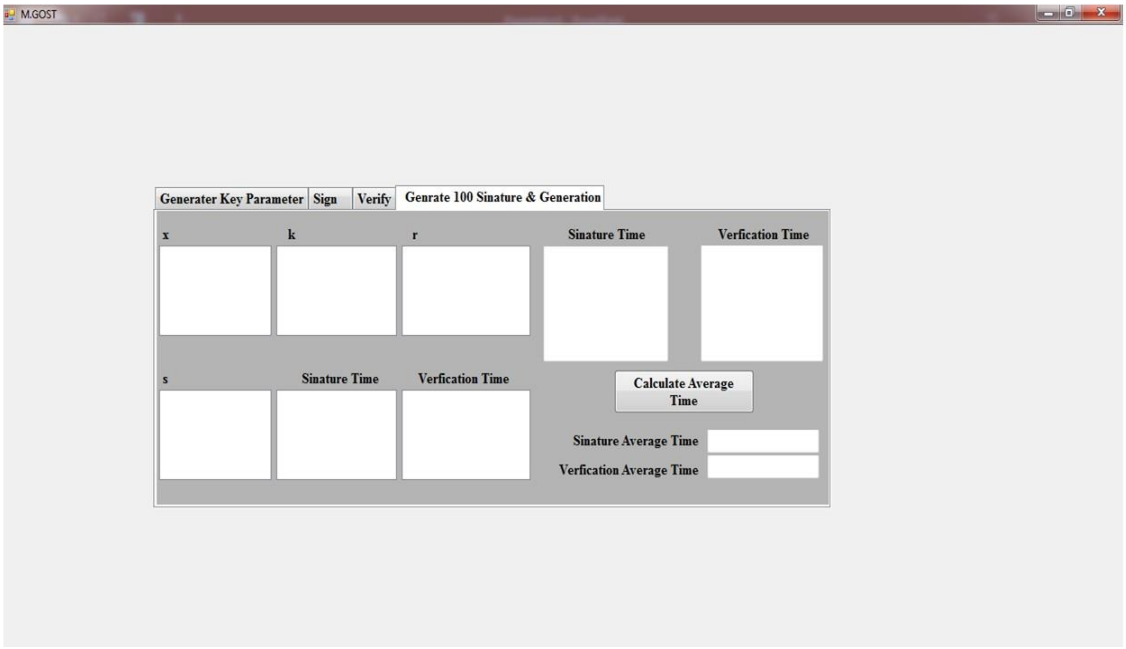

GOST
Generater Key Parameter
Sign
Verify
Genrate 100 Sinature & Generation
x
k
r
Sinature Time
Verfication Time
s
Sinature Time
Verfication Time
Calculate Average Time
Signature Average Time
Verfication Average Time

2- M.GOST Algorithm






M.GOST
Generater Key Parameter
Sign
Verify
Genrate 100 Sinature & Generation
p
3023
q
1511
g
y
M
0< H(M) < q
r
S
Verify
z1=(s *r^-1) mod q
v=(a^z1*y^z2 mod p) mod q
z2=-r*h^-1 mod q
Time
The Signature is


M.GOST
Generater Key Parameter
Sign
Verify
Genrate 100 Sinature & Generation
x
k
r
Sinature Time
Verfication Time
s
Sinature Time
Verfication Time
Calculate Average Time
Sinature Average Time
Verfication Average Time

Appendix B

كتاب الاستلال

عمادة الدراسات العليا والبحث العلمي
Deanship of Graduate Studies & Scientific Research

2018/05/09

Enhancement of Digital Signature Scheme

تعزيز نظام التوقيع الرقمي

Prepared by

Ruqa Abdulkareem Salih Al-shenawa

Supervisor: Prof. Hamza A. Al-Sewadi

نسبة الاستلال: 31 %

عمادة الدراسات العليا والبحث العلمي