ZOT-BINARY: A NEW NUMBER SYSTEM AND ITS APPLICATION ON NUMBER THEORY BASED PUBLIC-KEY CRYPTOGRAPHY

by

SHAHRAM JAHANI

**Thesis submitted in fulfillment of the
requirements for the degree of
Doctor of Philosophy**

January 2016

ACKNOWLEDGEMENTS

First of all, there is no word to show my deeply grateful to the Almighty God for his mercy toward me and his blessing for me to complete this thesis. It is my great pleasure to express my sincere gratitude to my supervisor Professor Dr. Azman Samsudin who generously dispenses his precious time to guide me in my thesis despite his busy schedule in the School of Computer Sciences. I'm very grateful being able to work under his intellectual, brilliant and strong supervision. I am thankful for his aspiring guidance, invaluably constructive criticism and friendly advice during my research work. I'd like to thank him for his trust on me and support my research through his grant. I would like to place on record, my sincere acknowledgement to the Dean and all staff members from the School of Computer Sciences, USM, for making all these possible. I would also like to express my deep sentiment to my family members especially to my dear wife and my son for their patience and encouragement. Thanks also go to all my friends in the research lab, who have made a sincere and helpful atmosphere of friendship in these years specially Amir Akhavan and Arash Eghdamian, who helped me when I needed in spite of being busy with their own work.

TABLE OF CONTENTS

CHAPTER THREE: PROPOSED NUMBER SYSTEMS, MULTIPLICATION AND EXPONENTIATION ALGORITHMS 58

CHAPTER FOUR: IMPLEMENTATION 90

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

BD	Big-Digit
Bd	Big-digit
BO	Big-One
BONS	Big-One Number System
Bo	Big-one
BT	Big-Two
Bt	Big-two
BDNS	Big-Digit Number System
BIN	Binary
CBONS	Canonical Big-One Number System
CBTNS	Canonical Big-Two Number System
CLBONS	Canonical Limited Big-One Number System
CLBTNS	Canonical Limited Big-Two Number System
DA	Double-and-Add
DBNS	Double-Base Number System
DES	Data Encryption Standard
DH	Diffie-Hellman
DLP	Discrete Logarithm Problem
DS	Digital Signature
DSA	Digital Signature Algorithm
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDLP	Elliptic Curve Discrete Logarithm Problem

ECDSA	Elliptic Curve Digital Signature Algorithm
Ind	Discrete Logarithm
KTNS	Koyama and Tsuruoka Non-Sparse
L2R	Left-to-Right
LBONS	Limited Big-One Number System
LBTNS	Limited Big-Two Number System
LUT	Look up-Table
LZOTB	Limited ZOT-Binary
MOF	Mutual Opposite Form
NAF	Non-Adjacent Form
PNR	Position Number System
R2L	Right-to-Left
RSA	Rivest-Shamir-Aldeman
ZOT	Zero-One-Two
ZOTB	ZOT-Binary
ZOTB-CLM	ZOT-Binary Classical Multiplication

ZOT-BINARY: SISTEM NOMBOR BAHARU DAN APLIKASINYA PADA SISTEM KRIPTOGRAFI KUNCI AWAM YANG BERASASKAN TEORI NOMBOR

ABSTRAK

Kriptosistem Kunci Awam telah digunakan secara meluas dalam protokol seperti pengurusan kekunci, pengesahan, penyulitan kekunci, dan lain-lain. Teori Nombor yang berasaskan Kriptosistem Kunci Awam adalah salah satu cabang utama dalam sistem Kriptografi Kunci Awam. Dua operasi utama dalam Teori Nombor berasaskan Kriptografi Kunci Awam adalah pendaraban dan nombor besar. Antara contoh kriptosistem yang terkenal yang mendapat manfaat daripada operasi ini ialah enkripsi dan dekripsi RSA, tandatangan digital EIGamal, dan pertukaran kunci Diffie-Hellman. Prestasi kriptografi primitif ini sangat bergantung pada kecekapan kedua-dua operasi tersebut. Adalah menjadi sesuatu kebiasaan untuk melakukan penambahbaikan terhadap kecekapan pendaraban dan pengeksponen melalui penggunaan kaedah pengekodan semula atau penggunaan sistem nombor bagi mengurangkan ukuran berat Hamming. ZOT adalah kaedah pengekodan semula yang terkini bagi mengurangkan ukuran berat Hamming. Tetapi, oleh kerana ZOT bukan berasaskan sistem kedudukan nombor, maka kos perlaksanaannya adalah tinggi. Fokus kajian ini adalah untuk membangunkan/mencipta ZOT yang cekap, berasaskan satu sistem kedudukan nombor yang mampu mempertingkatkan prestasi asas algoritma bagi pendarapan dan pengeksponen. Satu daripada sistem kedudukan nombor yang dicadangkan dalam kajian ini adalah *ZOT-Binary* yang dapat mengurangkan berat Hamming secara siginifikan. Seterusnya, cadangan sistem kedudukan nombor lain adalah *LZOT-Binary* iaitu satu sistem yang dapat mengurangkan saiz *look-up table* berbanding dengan *ZOT-Binary*. Untuk menilai keberkesanan sistem nombor yang dicadangkan, suatu persekitaran yang dapat mengendali nombor yang besar telah dibangunkan terhadap perningkatan operasi pendaraban, algorithma kuasa dua dan pengeksponenan masing-masing, Operasi yang dipertingkatkan telah dibandingkan dengan operasi asal. Berdasarkan dapatan kajian ini, ukuran pemberat Hamming *ZOT-Binary* dan *LZOT-Binary* adalah masing-masing 22 dan 23 peratus. Hasil kajian menunjukkan bahawa algoritma pendarapan klasik berasaskan *ZOT-Binary* iaitu *ZOTB-CLM* bagi nombor dalam julat 128 bit hingga 8 Kbit adalah hampir 20 kali lebih cepat daripada yang asal. Pada nombor dalam julat yang sama iaitu 128-bit kepada 8-Kbit, algoritma pendarapan yang dicadangkan, *ZOTB-CLM* menunjukkan kecepatan 5 hingga 25 kali lebih daripada algoritma pendarapan *Karatsuba-klasik*. Dapatan kajian terhadap peningkatan kuasa dua ZOTB-SQ menunjukkan bahawa algoritma ZOTB-SQ juga signifikan iaitu 41 hingga 53 kali lebih cepat bagi pengeksponenan nombor dalam julat 128-bit hingga 8-Kbit, berbanding dengan algoritma pengeksponenan klasik. Dapatan juga menunjukkan bahawa algoritma pengeksponenan berdasarkan *LZOT-Binary* adalah lebih kurang 36 kali lebih cepat daripada algoritma pengeksponenan yang terkenal bagi nombor 128-bit dan berkurangan kepada 9.6 kali bagi nombor 8-Kbit. RSA telah dipilih sebagai kajian kes bagi menunjukkan keberkesanan sistem nombor yang dicadangkan oleh kajian ini dan operasi peningkatan terhadap kecekapan Teori Nombor berasaskan Sistem Kriptografi Kunci Awam. Jumlah masa pelaksanaan bagi algoritma RSA yang beroperasi dengan menggunakan sistem nombor *LZOT-Binary* adalah lebih kurang 10 hingga 36 kali lebih cepat berbanding algoritma yang asal untuk julat nombor antara 128 bit hingga 8 Kbit. Sebagai rumusan, dapatan kajian tesis ini menunjukkan bahawa sistem nombor yang dicadangkan dan algoritma yang dipertingkatkan adalah sangat sesuai untuk digunakan pada Sistem Kriptografi Kunci Awam yang berasaskan Teori Nombor.

ZOT-BINARY: A NEW NUMBER SYSTEM AND ITS APPLICATION ON NUMBER THEORY BASED PUBLIC-KEY CRYPTOGRAPHY

ABSTRACT

Public-key cryptosystems are widely used in protocols such as key agreement, authentication, encryption; etc. Number theory based Public-key cryptosystems are one of the main branches in public-key cryptosystems. The two main operations in number theory based public-key cryptography are large number multiplication and exponentiation. For RSA encryption and decryption, ElGamal digital signature and Diffie-Hellman key exchange are some of the well-known example of these cryptosystems which benefit from these operations. The performance of these cryptographic primitives is highly dependent on the efficiency of these operations. Improving the efficiency of multiplication and exponentiation through the use of a recoding method or utilizing a number system which can decrease the Hamming weight of numbers is very common. ZOT recoding method is one of the latest recoding methods used to decrease the Hamming weight of numbers. However, since it is not positional number systems its cost is high. The focus of this study is to devise an efficient ZOT-base positional number system capable of improving the performance of multiplication and exponentiation-based algorithms. One of the proposed positional number systems in this study *is ZOT-Binary* with the result of a significant reduction in the Hamming weight. *LZOT-Binary* is another proposed positional number system with a reduced size look-up table compared to *ZOT-Binary*. To evaluate the efficiency of the proposed number systems and their respective enhanced multiplication, squaring and exponentiation algorithms, an environment that can handle large numbers was designed. The enhanced operations were compared to the original ones. Based on the results of this study, the Hamming weights of *ZOT-Binary* and *LZOT-Binary* are 22 and 23 percent respectively. The results show that classical multiplication algorithm based on *ZOT-Binary*, *ZOTB-CLM*, is approximately 20 times faster than the original, for numbers within the range of 128 bits to 8 Kbits. For the same 128-bit to 8-Kbit numbers, the proposed multiplication, *ZOTB-CLM,* is 25 to 5 times faster than the Karatsuba-classical multiplication algorithm. The results on the enhanced squaring, ZOTB-SQ, indicate that ZOTB-SQ is also significantly faster, that is 53 to 41 times faster for the 128-bit to 8-Kbit numbers, than the classical squaring algorithm. The findings also show that the exponentiation algorithm based on *LZOT-Binary* is about 36 times faster than the popular exponentiation algorithm for 128-bits numbers and decreases to 9.6 times faster for 8-Kbits numbers. To show the impact of the proposed number systems and the enhanced operations on the efficiency of number theory based public-key cryptosystems, RSA was chosen as a case study. The total execution time for RSA algorithm that runs on *LZOT-Binary* number system is about is about 36 to 10 times faster than the original algorithm for numbers ranging from 128 bits to 8 Kbits. In summary, the findings from this research indicate that the proposed number system and the enhanced algorithms are highly suitable for existing number theory based public-key cryptosystems.

CHAPTER ONE: INTRODUCTION

1.1 General Overview

The steady acceleration of advances in computer science, over the past few decades have resulted an insatiable human appetite for greater data efficiency and convenience. Thus, cryptography has become a part of daily life as a branch of computer science that provides security over data. Secure communications, financial transactions, education, healthcare are just few examples of the many elements of modern society that are closely in interaction with cryptography. This tremendous development of information technology has brought up importance of information security more than ever [1, 2]. According to [1, 2], the main responsibilities of cryptography in the information security are about achieving the following goals: Confidentiality, Data integrity, Authentication and Non-repudiation. In order to achieve the above-mentioned goals, some cryptographic primitives have been designed which can be listed as follows and detailed out in Figure 1.1.

Unkeyed primitives: These primitives are not based on any keys. Hash functions and random sequence generators are two main primitives in this class.

Symmetric-key primitives: In this class, a single key called secret key is shared between two parties. These primitives are used to encrypt a message, authenticate sender and data integrity. Some of the most well-known symmetric ciphers are: DES [2], AES [3] and RC5 [4].

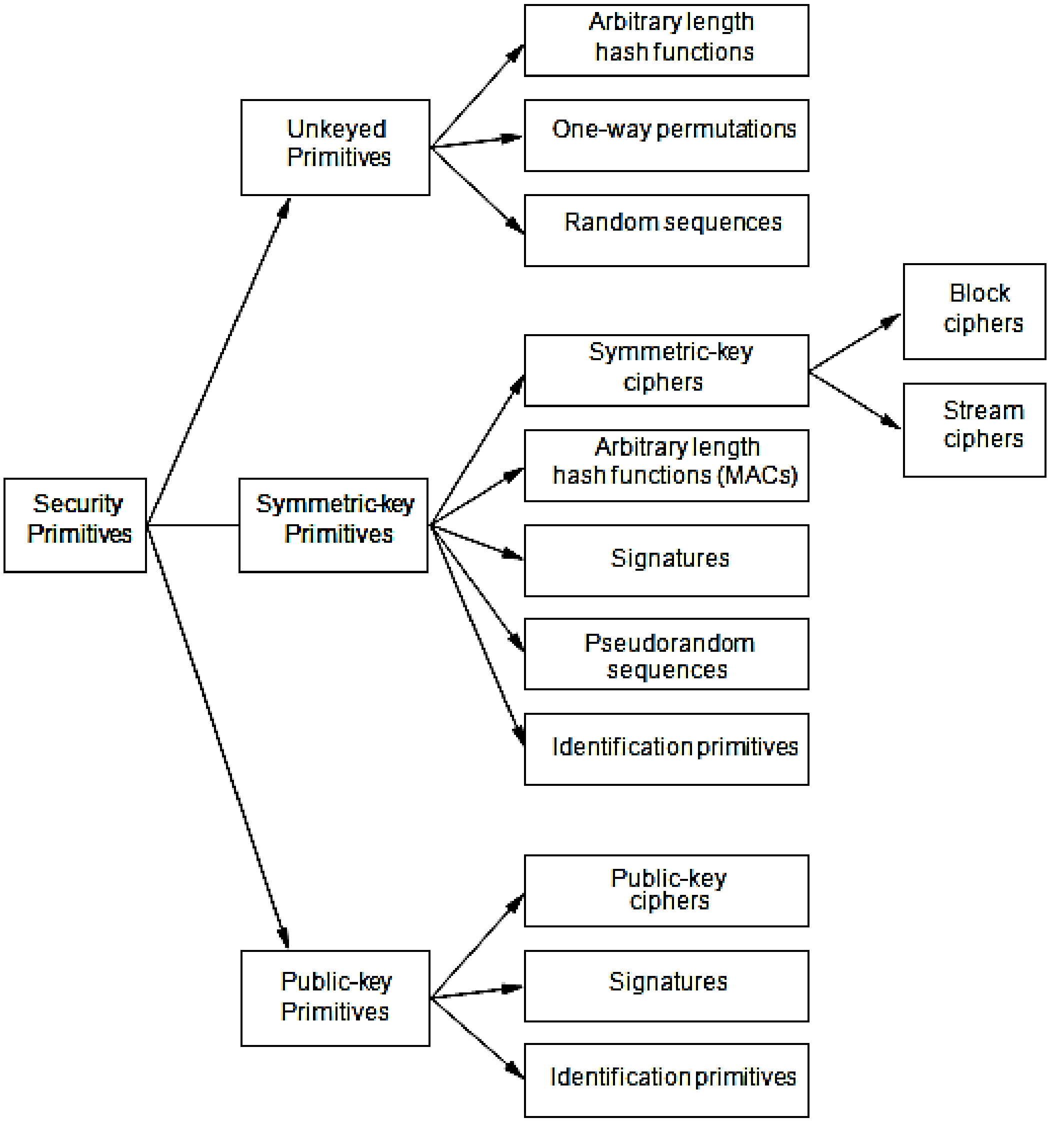

Figure 1.1: A taxonomy of cryptographic primitives [1]

Asymmetric-key (Public-key) primitives: This primitive is also known as public key, uses two mathematically related keys: public key and private key. Practically, it is infeasible to extract one key with just knowing the other key. This specification of public key primitives enables the users to transmit information over unsecure channels, while the key in the symmetric key schemes must be transmitted through a secure channel.

All four cryptographic goals mentioned before could be obtained using public key cryptosystems, while the symmetric key schemes do not include the non-repudiation characteristics. The reason is the existence of the exclusive private key for each user that is not shared with other users.

Generally, public key cryptosystems are slower than symmetric key cryptosystems [5]. The operations used in the current defector public key cryptosystems are algebraic operations, while in symmetric key the operations are logical operations, which perform faster on computer [5]. Consequently, it is preferred that public key cryptosystems be used for applications which deal with small messages. Digital signature and key exchange schemes are also instances of these applications. In fact, symmetric and asymmetric cryptosystems are two complement cryptosystems and hybrids of them can meet all of the cryptographic goals [3].

During the recent decades, number theory and public key cryptosystem have become intertwined more and more. Diffie and Hellman, introduced the first key exchange scheme in 1967 based on modular exponentiation [6]. Few years later, in 1978 one the most used public key cryptosystem, called RSA [7], was introduced by Rivest, Shamir and Adleman. The main operation in RSA cryptosystem is also based on the modular exponentiation. ElGamal key exchange [8] is another example of public key cryptosystems, which has been developed based on the modular exponentiation. Consequently, more efforts in public key cryptography have been dedicated to find algebraic operations or one-way functions that meet the specifications of public key cryptography [1].

The performance of a public key cryptosystem is strongly related to the operations used in the cryptosystem. This research focuses on the modular exponentiation and multiplication, which are the two fundamental operations in the current public key cryptography.

Modular exponentiation: This operation is a one-way function used in public key cryptosystems such as RSA encryption, Diffie-Hellman key exchange and ElGamal digital signature. Modular exponentiation expression is normally written as follows:

$$x = a^{b} \bmod n, \tag{1-1}$$

However, finding the inverse of modular exponentiation requires solving the discrete logarithm equation,

$$b = Ind_{a}^{x} \tag{1-2}$$

which is a known hard problem [9]. It would take far too long and practically impossible when the case is, computing an RSA signature with a 1024-bit key that is equivalent of computing (a^{1024}) modulo of a large number.

The naive approach to calculate the modular exponentiation is just a consecutive multiplications and divisions [1, 11, 12], which is not an efficient way for large exponent.

$$a^{b} \bmod n = (\overbrace{a \times a \times \ldots \times a}^{b}) \bmod n, \tag{1-3}$$

Binary exponentiation is a more efficient method with fewer operations. This method is based on the binary representation of the exponent. If:

$$b = \sum_{i=0}^{k-1} (b_i 2^{i}), \tag{1-4}$$

where $b_i \in \{0,1\}$, then

$$a^{b} = \prod_{i=0}^{k-1} \left(\mathrm{a}^{2^{i} b_i}\right). \tag{1-5}$$

The term $a^{2^{i}}$can be obtained by squaring the ith term: $a^{2^{i-1}}$. The number of operations for calculating a^{b} in naïve method is $(b-1)$ multiplications while in binary method, result can be computed by k times of squaring and $\frac{k}{2}$ times of multiplications (on average). From Equation 1.4, it can be concluded that $k = log_2 b$, in other words, the number of operations in binary method is considerably lower than the naïve method. Since the algorithm of the binary method starts from the least significant bit upwards to the most significant bit, it is also called Right-to-Left (R2L) binary method. Further improvement in usage memory could be achieved by starting binary algorithm from the most significant bit towards to the least significant bit, called Left-to-Right (L2R) binary method [10].

Multiplication is one of the main operations in exponentiation calculation and it is slower than squaring by a small factor [22]. Multiplication can be time consuming when used frequently to multiply large numbers. Another approach to improve the time efficiency of binary exponentiation algorithms is decreasing the number of multiplication operations. The *m*-ary exponentiation [11, 12] works based on this idea. The exponent is segmented into *m*-bit groups. All of the values of a^g for $0 < g < 2^m$ are computed earlier and saved in a table called look-up table (LUT). Suppose the length of number and the number of segments are *k* and *t*, then the number of multiplication and squaring would be $t = \frac{k}{m}$ and *k* respectively. It should be considered that there is an optimum value for *m* based on the value of *k*, the length of number [13].

Sliding window exponentiation is a modification of *m*-ary method. In this improved modification, only the upper half of the look-up table in the range of $2^{m-1} < g < 2^m$ should be created. The optimum value of *m* (window size) against of *k* (length of number) has been reported in [13]. Each segment in sliding window starts by *1*. Following example shows the difference.

m-ary method : $A = \underbrace{11}\ \underbrace{100}\ \underbrace{101}\ \underbrace{000}\ \underbrace{101}\ \underbrace{111}\ \underbrace{000}\ \underbrace{000}\ \underbrace{011}$

Sliding Window: $A = \underbrace{111}\ 00\ \underbrace{101}\ 000\ \underbrace{101}\ \underbrace{111}\ 0000000\ \underbrace{11}$

There are also some studies on improving exponentiation algorithm for special cases of exponentiation. For example, a survey on fixed base exponentiation algorithms has been reported in [14, 15]. Exponentiation with known exponent has been also studied by some other scholars [16].

The above mentioned methods to improve exponentiation computation are mostly focused on reducing the number of sub-operations of multiplication used in the exponentiation algorithm. There is another approach that concerns on using fast multiplication and squaring as main sub-operations in exponentiation. Following, the most important studies related to this approach are briefly reviewed.

Modular Multiplication: There are two ways to calculate modular multiplication: multiplying and then reducing, or combining the multiplication and the reduction steps together as an operation. Therefore, any algorithm, which improves the performance of one or more of these operations, can be used to enhance the performance of modular multiplication.

The most popular multiplications algorithms for large numbers are classical [11], Karatsuba [17], Toom-Cook [18, 19] and Shonang-Strassen [20] algorithms; in which the first two algorithms are more common than the others and the most used algorithms in the range of the currently used public-key cryptosystems [21].

Although the complexity of the classical multiplication algorithm,$O(n^2)$, is higher than the complexity of the other algorithms, Xianjin and Longshu [22] have shown that the classical multiplication algorithm is theoretically efficient for multiplying numbers that are shorter than 255 digits. In addition, it has also been shown that the classical multiplication algorithm is efficient in terms of memory utilization. The Karatsuba multiplication algorithm has less complexity, $O(n^{1.58})$, compared to the classical multiplication algorithm. However, the overhead of the Karatsuba multiplication algorithm for lower-range numbers (i.e., fewer than 255 digits) has caused researchers [22] to combine both the classical and Karatsuba algorithms to achieve better overall algorithm efficiency.

Another approach to improve the multiplication algorithms is creating or using alternative positional number systems [23-25]which are computationally more efficient than non-positional number systems [26]. For example, by proposing signed-binary numbers [23] instead of the usual standard binary numbers, some researchers have managed to decrease the number of partial products in the classical multiplication algorithm and have therefore increased the overall algorithm efficiency. In positional number systems, an integer of radix-*r* can be written as follows,

$$(a_n \dots a_2 a_1 a_0)_r = a_n r^n + \cdots + a_2 r^2 + a_1 r^1 + a_0 \qquad (1\text{-}6)$$

If $0 \leq a_i < r$; then this representation is unique. We call a_i as a digit, and its related set, such as $S = \{0,1,\dots,r-1\}$ as the digit set. Decimal numbers with $r = 10$ and $a_i \epsilon\, S = \{0,1,\dots,9\}$, as well as the binary numbers [11] with $r = 2$ and $a_i \epsilon\, S = \{0,1\}$, are the two most well-known positional numbering systems. In this work, we focus on the binary positional number systems.

There is another type of binary numbers known as Signed Binary (SB) representation with $S = \{0, \pm 1\}$. The Booth [23], higher-radix Booth [27, 28], Non Adjacent-Form NAF [29] and Mutual Opposite Form MOF [24] are examples of SB number systems that have been invented to reduce the number of sub-operations in arithmetic calculations such as multiplication [23, 30], exponentiation [11, 12] and scalar multiplication [24, 29] computations. Multi-Based Number System (MBNS) is another group of number systems [31], which uses more than one base. The Double-Base Number System (DBNS) [25] is an example of a number system from this group. The numbers in DBNS are represented as follows,

$$\sum c_i 2^{a_i} 3^{b_i} \qquad (1\text{-}7)$$

where $c_i \epsilon \{0,1\}$ and $a_i, b_i \epsilon Z^+$.

Improving the efficiency of multiplication algorithms by using LUT (Look-up tables) is another method that researchers try to benefit from. For example, a group of work based on the binary numbers is the sliding window method, namely, wNAF [32, 33], wMOF [24] and wmbNAF [34-36]. In summary, by using efficient numbers recoding, both number of calculations and efficiency of each single calculation will be improved. The study of number systems and exploring this area to enhance or create new concepts could be remarkable for researchers.

1.2 Research Motivation

In the recent years, a number representation called ZOT [37, 38] which derived from the binary number system has been proposed and researchers showed that the performance of Karatsuba multiplication algorithm can be enhanced by using this recoding. This improvement is gained from the low Hamming weight property of the ZOT representation.

As mentioned earlier reducing the Hamming weight of the exponent, reduces the number of sub-operations in an exponentiation calculation. In addition, the lower Hamming weight of numbers in a multiplication, the less number of sub-operations in multiplication calculation. Thus, a number representation with lower Hamming weight can benefit exponentiation calculation from these two mentioned aspects.

Since ZOT representation is not a positional number system therefore, the overhead for applying it in a multiplication is relatively high. Considering this point leads to this work, that is to create a new number system based on ZOT representation. This number system should keep the advantages of ZOT recoding plus advantages of positional number systems in calculations.

1.3 Problem Statement

Modular exponentiation is one of the main operations used in number theory based cryptosystems which is a time consuming calculation by nature. Manipulating and recoding the exponent is a common method used by researchers to improve the efficiency of modular exponentiation. Using faster multiplication and squaring algorithms is also another approach to improve the efficiency of this operation. Recoding numbers or changing the representation of numbers is the most common method used to make multiplication and squaring more efficient. A recording method of binary numbers called ZOT is one of the latest efforts to improve the multiplication for public-key cryptosystems via decreasing the Hamming weight of the numbers. Since ZOT representation is not a positional number system, it cannot be easily generalized and used for the other multiplication algorithms or squaring and exponentiation calculations, the cost of such calculations would be relatively high anyway. Therefore, the aims of this research are first, to create positional number systems based on ZOT representation to apply in multiplication and squaring algorithms; and second, to use the new number systems to recode an exponent of an exponentiation operation to achieve a lower Hamming weight exponent and consequently less calculation, therewith, improving the efficiency of number theory based public-key cryptosystems such as RSA.

Research Questions: The following research questions are retrieved from problem statement:

1. Can the number recoding method called ZOT which efficiently decrease the HW of numbers be generalized to an efficient positional number system for calculations such as multiplication, squaring and exponentiation?

2. How efficient would be the modular operations such as multiplication, squaring and exponentiation after being enhanced by the proposed positional number systems in range of number theory based public-key cryptosystems?

3. How efficient would be the number theory based public-key cryptosystems after applying the new positional number systems and the new enhanced operations?

1.4 Research Scope

Authentication, Key agreement and key transport are some of the popular protocols in information security that have been developed base on the public key primitives. Many operations in modular arithmetic have been used in public key cryptosystems such as RSA, Diffie-Hellman (D-H) key exchange and ElGamal digital signature.

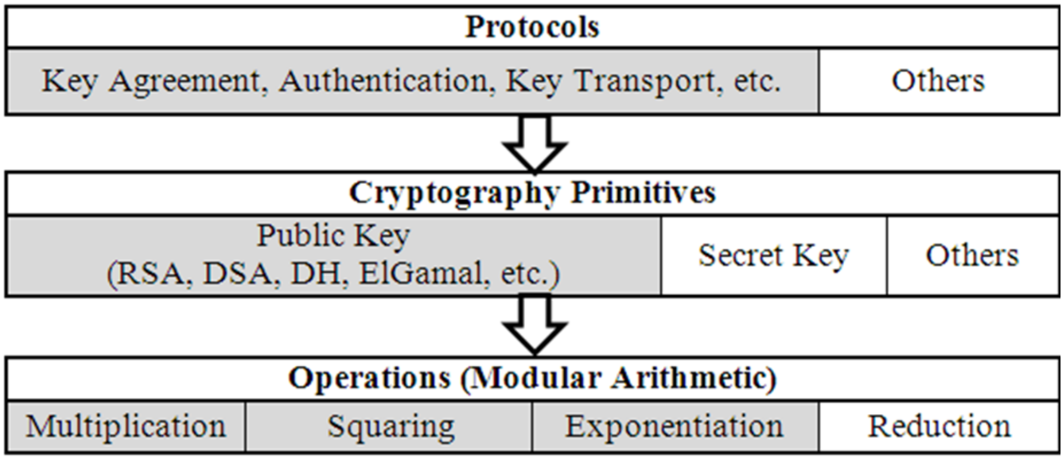
Protocols
Key Agreement, Authentication, Key Transport, etc.
Others
Cryptography Primitives
Public Key
(RSA, DSA, DH, ElGamal, etc.)
Secret Key
Others
Operations (Modular Arithmetic)
Multiplication
Squaring
Exponentiation
Reduction

Figure 1.2: Scope of research

This study focuses on some of these operations such as multiplication, squaring and exponentiation. Since the concentration of this research is on the number theory based cryptosystems, the performance of these operations in range of public-key cryptosystems has been investigated. The scope of this study will be limited to methods of improving the running time of these operations (see Figure 1.2). Since the classical division does not depend on the Hamming weight or other special shape of numbers [39], then it is not in the scope of this research.

1.5 Research Objectives

The overall goal of this research is to promote ZOT recoding method for calculations such as multiplication, squaring and exponentiation in range of number theory based public-key cryptosystem. The main objectives of this thesis are listed as below:

- To define number systems based on ZOT recoding and explore the potential of the proposed number systems on improving the time efficiency of the number theory based public-key cryptosystems.
- To enhance multiplication, squaring and exponentiation algorithms, the highly used operations in number theory based public-key cryptosystems, based on the proposed number systems.
- To apply the proposed number systems and the enhanced computational algorithms on RSA (as a case study) to improve its time efficiency as a representative of number theory based public-key cryptosystems which use multiplication, squaring and exponentiation.

1.6 Research Methodology

To meet the three research objectives mentioned in Section 1.5, the following method, experiments and tools have been set up.

The review of modular arithmetic, which has a crucial role in public key cryptosystems showed us that designing an efficient number system is the most common way to improve the efficiency of the multiplication, squaring and exponentiation algorithms. Hence, in the first step, few positional number systems based on ZOT recoding were defined. The literature review indicates this recoding method has a good potential to be a number system and improve the efficiency of multiplication. To enhance the currently used multiplication, squaring and exponentiation algorithms in number theory based public-key cryptosystems the proposed number systems were used and the enhanced operations were applied on RSA cryptosystem. This system is one of the well-known cryptosystems in number theory based cryptosystems, which is used as standard in many applications. In addition, since the operations used in RSA are as same as operations used in other cryptosystems such as ElGamal and Diffie-Hellman, it can demonstrate the impact of the proposed number systems and enhanced operations on efficiency of number theory based cryptosystems. To evaluate and analyze the proposed number systems and operations the following experiments have been set up. The tools and conditions of experiments are also explained herewith.

1**. Analyzing the efficiency of the proposed number system to improve the efficiency of the proposed enhanced operations:** The first metric used for measuring the efficiency of the proposed number systems is Hamming weight. The Hamming weight of the proposed number systems is calculated in two ways; mathematically and experimentally. The lower Hamming weight is, the more potential there is for efficiency. The second metric is the running time of converting numbers from binary to the proposed number systems. It would be contributory to measure and compare the cost of conversion of the proposed number systems to binary and the running time of the proposed operations to determine which one of the number systems is the more efficient in enhancing the operations of multiplication, squaring and exponentiation.

2. Analyzing the efficiency of the enhanced operations: Three metrics have been considered to evaluate the efficiency of these operations. The first one is to determine the number of non-zero partial products of the algorithms that is directly related to the HW of the original numbers and is a more reliable metric when the main sub-operations are partial products. The second metric is the running time of the proposed operations. To investigate the more efficient number system among the proposed number systems for improving the efficiency of the enhanced operations, the running time of the enhanced operations after applying the proposed number systems is measured. Since the multiplication and squaring operations are two main operations in exponentiation, then identifying the more efficient multiplication and squaring helps to define the more efficient exponentiation as well. Finding the more efficient exponentiation guides to know how to maximize the improvement of the efficiency of RSA by choosing the more efficient sub-operations. The last metric is measuring the memory for the LUTs required for multiplication, squaring and exponentiation. Depending on the application, this metric helps us choose the appropriate number system for enhancing the operations.

3. **Analyzing the efficiency of RSA after applying the enhanced operations:** This cryptosystem is one of the well-known cryptosystems in number theory based cryptosystems, and is used as standard in many applications. In addition, since the operations used in RSA are the same operations as used in other cryptosystems such as ElGamal and Diffie-Hellman, it can readily demonstrate the impact of the proposed number systems and its enhanced operations on efficiency of number theory based cryptosystems. The main metric to measure the efficiency of RSA after applying the enhanced operations is counting the number of operations, especially the number of partial products. The second metric is running time which helps us evaluate the overall efficiency of RSA, experimentally. The last metric used to analyze is the amount of memory for saving the required LUTs.

Tools and Conditions:

Software: In order to collect data and analyze the proposed number systems and operations, a computer software supporting large number calculations has been developed. This software includes all the proposed algorithms and operations plus the original ones. It also consists of the required functions to measure the different metrics that have been considered for evaluation of the proposed algorithms.

Hardware: In order to be more confident with the experimental results, all experiments have been conducted using two PCs. The results are then averaged.

Data sampling: The number of samples has been calculated and the data have been generated randomly within the range of public-key cryptography.

1.7 Research Contributions

The significant contributions of this research are as follows:

- New positional number systems based on ZOT recoding:
    - CBONS, LCBONS, ZOT-Binary, LZOT-Binary.
- Enhancing classical multiplication, squaring and Left-to-right exponentiation algorithms for large numbers in range of number theory based public-key cryptosystems:
    - ZOTB-CLM, LZOTB-CLM, LZOTB-SQ, LZOTB-L2REXP-E, LZOTB-L2REXP-M, LZOTB-L2REXP-EM,
- Applying the new number systems and enhanced operations on RSA:
    - LZOTB-RSA

1.8 Thesis Outline

This thesis has been organized into six chapters. The first chapter provides an overview of the research content. Chapter 2 continues with the related works on the large number multiplication and exponentiation as the most frequently used operations in number theory based cryptosystems. Chapter 3 proposes few new number systems based on ZOT recoding method, such as ZOT-Binary, LZOT-Binary, and enhances multiplication and exponentiation algorithms in range of number theory based public-key cryptosystems by using the proposed number systems. Details on the implementation of the conversion algorithms, such as converting binary numbers to ZOT-Binary numbers, and all the enhanced algorithms are presented in Chapter 4. Meanwhile, the results of this research are presented in the Chapter 5. Chapter 5 provides the results and discussion; and finally Chapter 6 concludes.

CHAPTER TWO: LITERATURE REVIEW

The history of human civilization shows that the numbering system has always been an important issue in human daily life. The huge number of applications related to numbers and arithmetic operations implies the signification of the numbering system as an essential necessity more than in the past.

Followed by the emergence of numbers, arithmetic operations were gradually invented and developed. The development between number systems and arithmetic operations has been bilateral. Based on this relevancy, researchers have improved arithmetic operations by inventing alternative numbering systems.

The main goal of this research is to increase the efficiency of large integers' multiplication and exponentiation which have many important applications, such as cryptography and other scientific calculations. To achieve this goal, modification of known multiplication and exponentiation algorithms running on top of a new numbering system has been identified as the approach.

This chapter continues with the general definitions used in this work. Sections 2.3 reviews the works related to the positional numbering system specially the related works in radix 2. We describe the integer arithmetic algorithms used in this research in Section 2.4. Section 2.5 is dedicated to the number theory based public-key cryptosystems and RSA cryptosystem. A summary of this chapter is presented in Section 2.6.

2.1 Definitions

In this section, the definitions frequently used in this work such as *Hamming weight* and *complexity* are explained. The other definitions will be defined in their related sections.

2.1.1 Hamming Weight

The *Hamming weight* (HW) of a string u (or a vector $u \in F^n$) is the number of non-zero symbols (or components $u_i \neq 0$) [40]. For a binary number or string of bits, Hamming weight is the number of 1's in the binary number or string. For example if $A = 1000001110$, then $\mathrm{HW(A)} = 4$. For a *n*-bit random binary number, the Hamming weight is always supposed $\frac{n}{2}$. In some work the percentage of Hamming weigh to the length of number is used.

2.1.2 Computational Complexity

An algorithm is a process to solve a class of mathematical problems on a computer. The complexity of an algorithm can be defined as the cost (usually the running time, storage, and any other relevant parameter) of solving the problems by the method specified in algorithm. The cost of solving some problems is very high therefore, researchers try to discover a faster way (a faster algorithm) to do them. Measuring the computational complexity of algorithms helps us to compare the algorithms.

There are five standard asymptotic notations that are used to show and compare the growth or complexities of algorithms. These notations are big-O (O) notation, little-o (o) notation, big omega (Ω) notation, little omega (ω) notation and big theta (θ) notation.

2.2 Number Systems

Although the origins of numbers are not very clear, it is still safe to say that the advancements of civilizations have revolved around numbers. In our daily and scholarly lives, the role of numbering systems is getting more significant and crucial than in the past. Early civilizations in 4000 B.C. utilized numbers just for counting and comparing. There was no possibility of calculation with primal numeral systems. Trading, seasonal-agriculture, and astronomy were main reasons to motivate the ancient people in Egypt and Greece to invent the numbering system with computational capabilities.

One of the most significant changes in the history of numbering systems started with positional number systems. The first achievement of positional number systems was to represent the large numbers easily; dealing with large numbers was one of the obstacles of non-positional number systems. Doing daily and complicated computations without positional number systems is impossible.

In the next sections, positional number systems are defined and the most popular of them such as decimal and binary are explained. The focus in this research would be on the binary positional number systems which have a significant role in computer algebra, computer architecture and some arithmetic operations.

2.2.1 Positional Number Systems

In *Positional Number Systems* (PNR), an integer of radix-*r* can be represented as follows:

$$(a_n \ldots a_2 a_1 a_0)_r = a_n r^n + \cdots + a_2 r^2 + a_1 r^1 + a_0 = \sum_{i=0}^{n} a_i \, r^i \qquad (2\text{-}1)$$

If $\mathrm{a_i} \in S = \{0, \dots, r-1\}$; then this representation is unique [41]. a_i and S are called as *digit*, and *digit set* respectively. And the notations a_n and a_0 are also called the *Most Significant Digit* MSD and the *Least Significant Digit* LSD correspondingly.

Two most well-known positional number systems [11] are:

- Decimal numbers: $S = \{0, \dots, 9\}$ and $r = 10$
- Binary numbers: $S = \{0,1\}$ and $r = 2$

Equation (2-1) only represents some of PNRs called *Fixed-base number system* (FBNS). Each term, $a_i r^i$, in the summation in this equation consists of two parts; digit a_i and $w_i = r^i$ which is called *weight* or *place value* of a_i. The weight of each digit is obtained by multiplying the weight of previous digit by base;

$$w_{i+1} = r \times w_i \tag{2-2}$$

The distinction between FBNS and *mixed-based number system* (MBNS) is the value of *r* in Equation (2-2). If r be constant like in Equation (2-1), we say it is a FBNS, while if this value is a variable we call it a MBNS. Equation (2-3) shows this property.

$$w_{i+1} = r_i \times w_i \tag{2-3}$$

where there are integers *i* and *j* such that $r_i \neq r_j$.

If we use the representation of Equation (2-4) to show a number by its digits and their corresponding weights

$$A = \begin{Bmatrix} digit \\ weight \end{Bmatrix} = \begin{Bmatrix} a_n \, a_{n-1} \\ w_n w_{n-1} \end{Bmatrix}, \dots, \begin{Bmatrix} a_0 \\ w_0 \end{Bmatrix}. \tag{2-4}$$

then FBNSs can also be represented by

$$\begin{Bmatrix} digit \\ weight \end{Bmatrix} = \begin{Bmatrix} a_n & a_{n-1} & & a_0 \\ rw_{n-1} & rw_{n-2}, & \cdots, & (w_0 = 1) \end{Bmatrix}, \tag{2-5}$$

and MBNS can be written as below:

$$\begin{Bmatrix} digit \\ weight \end{Bmatrix} = \begin{Bmatrix} a_n & a_{n-1} & & a_0 \\ r_{n-1}w_{n-1} & r_{n-2}w_{n-2}, & \cdots, & (w_0 = 1) \end{Bmatrix}. \tag{2-6}$$

Knuth [11] has been using the following representation for positional number systems.

$$(a_n \dots a_2 a_1 a_0)_{\{r_n, r_{n-1}, \dots, 1\}} = \begin{Bmatrix} digits \\ radices \end{Bmatrix} = \begin{Bmatrix} a_n & a_{n-1} & & a_0 \\ r_n & r_{n-1}, & \cdots, & (r_0 = 1) \end{Bmatrix} \tag{2-7}$$

where the weight of digit can be obtain by Equation (2-3). There are also some non-standard number systems with non-regular bases such as negative bases [42], fractional bases, real bases [43, 44], complex bases[43] and quadratic bases [42].

2.2.1.1 Binary Number System

Although some researchers believe that the binary number has been invented initially in the fifth century B.C by Pingala [45], Gottfried Leibniz is known as a mathematician who documented the modern system of binary numbers in 1703 [46, 47]. The lengthy representation of binary numbers was a large barrier to their daily utilization, but finally it has become a fundamental part of human life with the introduction of computer. In 1854 British mathematician George Boole introduced a new algebra based upon this number representation called *Boolean algebra* [48]. Computer science and digital systems have been developed based upon this algebra. Boolean algebra also plays a significant role in other topics such as number theory, statistics and set theory.

We showed earlier in Section 2.2.1, binary number system is a positional number system with digit set $S = \{0,1\}$ and base $r = 2$. Representation of numbers in this base is more lengthy compare to decimal numbers. For example:

$$1000000_{10} = 11110100001001000000_2.$$

Each digit in binary system is called *bit* and the *length of a number A* in *bits*, given by the formula

$$L(A) = L(A, 2) = \lfloor \log_2 A \rfloor + 1. \qquad (2\text{-}8)$$

2.2.1.2 Signed-Binary Number System

Efficiency of some of the algorithms in number theory such as multiplication and exponentiation depends on the Hamming weight of the binary numbers. To decrease the Hamming weight of binary numbers, researchers proposed new number systems derived from binary number system [23]. One of these number systems is called *Signed-binary* (SB) system which started by using '-1' in the representation of a binary number.

The idea behind of these kinds of number systems comes from the following series expansion in number theory;

$$\left(\overbrace{1 \dots 1}^{n+1}\right)_2 = \sum_{i=0}^{n} 2^i = 2^{n+1} - 1 = \left(1\,\overbrace{0 \dots 0}^{n}\,\bar{1}\right)_2, \qquad (2\text{-}9)$$

where $\bar{1} = (-1)$.

Hamming weight of each *n*-bit sequence of symbol '1' decreases from *n* to 2. For example:

$$(1111111)_2 = (1000000\bar{1})_2,$$

where

$$HW(1111111)_2 = 7 \quad \text{and} \quad HW(1000000\bar{1})_2 = 2.$$

The digit set and base in signed-binary number systems are defined as below

$$S = \{0,1,-1\} \quad \text{and} \quad r = 2. \qquad (2\text{-}10)$$

The representation by signed-binary is not unique. Booth [23], NAF [29] and MOF [24] representations are the most known methods of this kind of representations but with different Hamming weight.

Booth algorithm: Booth [23] in 1951 introduced an elegant algorithm to speed up the multiplication algorithm on digital processors. His method is known as the origin of the signed-binary representation of a number. He modified the common method of *add and shift* in the multiplication algorithm by a method that scanned the multiplicand and determined to decide to do addition, subtraction or nothing then shifted the result [49]. Although he did not directly represent the numbers in signed-binary number, the hidden idea in his method was the signed-binary idea.

Algorithm **(2.1**) describes the process of this Booth recoding. Let $A = (a_{n-1}, \dots a_0)$ and $B = (b_{n-1}, \dots b_0)$. Let $a_{-1} = 0$. We scan A from right to left for finding two adjacent bits $a_i a_{i-1}$ in the form of "01" or "10". If $a_i a_{i-1} = \text{"01"}$, then b_i is set to 1. Where $a_i a_{i-1} = \text{"10"}$, then b_i is set to '-1'. The rest of b_i's will be remain zero.

```
Algorithm (2.1)   : Booth Recoding
Input             : $A = (a_{n-1}, \dots a_0)$ is binary number
Output            : $B = (B_{n-1}, \dots b_0)$ is signed-binary number

1.    For $i = -1$ up to $n - 1$ do                          // Scan A from right to left
                                                              // Assume $a_{-1} = 0$
2.        a.  If $a_i = 0$ and $a_{i-1} = 1$ then $b_i = 1$   // Check $a_i a_{i-1}$ for" 01" or
          b.  If $a_i = 1$ and $a_{i-1} = 0$ then $b_i = -1$  // "10". Set $b_i$
3.    Return B
```

Example as shown by Equation (2-11) shows how Booth algorithm works.

$$A = \overbrace{111}^{100\bar{1}}\,000\,\overbrace{01111}^{1000\bar{1}} \;\rightarrow\; B = 100\bar{1}0001000\bar{1} \qquad (2\text{-}11)$$

In the above example, the number of non-zero digits decreased from HW(A) = 7 to HW(B) = 4. But, this does not always happen. Specially, when we have pairs of “01” or “10’. The following example makes this issue more clear.

$$A = 10101 : HW(A) = 3 \;\rightarrow\; B = 1\bar{1}1\bar{1}1\bar{1} : HW(B) = 7 \qquad (2\text{-}12)$$

Since in Booth algorithm two bits are scanned, it sometimes is called Booth 2. Researchers proposed Booth 3, Booth 4 and higher [27, 28, 50] to reduce the Hamming weight of binary numbers which shown in Example (2-12).

Non-Adjacent Form Recoding (NAF): The *Non-Adjacent Form* (NAF) recoding was proposed by Reitwiesner in 1960 [29]. As Algorithm **(2.2)** shows the NAF representation of a number which can be obtained by scanning the integer from right to left.

```
Algorithm (2.2) : NAF Recoding
Input           : $A = (a_{n-1}, \dots a_0)$ is binary number
Output          : $B = (b_n, \dots b_0)$ is signed-binary number (NAF)

1.  While $A > 0$
2.     For $i = 0$ up to $n - 1$ do                    // Read $a_i$ from right to left
3.        a. If $a$ is odd then do
                i.   $b_i = 2 - (A \ mod \ 4)$
                ii.  $A = A - a_i$
          b. Else
                i.   $b_i = 0$
                ii.  $A = \frac{A}{2}$
                iii. $i = i + 1$
4.  Return B
```

The following example shows a binary number and its NAF representation:

$$\begin{aligned} A = 1\,\overbrace{01111}^{1000\bar{1}}\,0000\,\overbrace{01111}^{1000\bar{1}} \;\rightarrow\; NAF(A) \\ = 101000\bar{1}00001000\bar{1} \end{aligned} \tag{2-13}$$

NAF representation assures that between two non-zero digit, there is at least a zero. The problem of Booth algorithm in its worst case has been solved in this algorithm.

Solinas [33] generalized the NAF recoding which can be used to improve the performance of EC computations, while a Left-to-right NAF recoding algorithm was proposed by Joye and Sung-Ming [51]. Darrel et al. [52] proved that every integer can be represented by NAF recoding uniquely. Morain et.al. [53] proved the average of Hamming weight of a number after NAF recoding would be minimal. Length of integer A, then $HW(A) \cong \frac{n}{3}$.

Mutual Opposite Form (MOF): Algorithm NAF (see Algorithm **(2.2)**) shows that the recoding method in NAF is done right to left, while left-to-right methods are more preferred for calculating exponentiation and EC multiplication [54]. The first left-to-right recoding algorithm called *Mutual Opposite Form* (MOF) introduced by Okeya et al. [24]. This method is bidirectional and its inventors showed that MOF representation of a number is unique [24]. The average Hamming weight of a number in MOF representation is about 50% (as same as of the binary representation) [24].

Although the Hamming weight of a MOF recoded number is greater than NAF recoded, researchers [24] proved that for applications such as EC multiplication, the total cost of MOF recoding and EC multiplication would be reasonable rather than of the NAF representation [29].