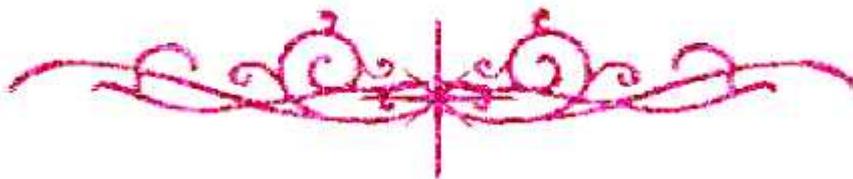




# بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ





# شبكة المعلومات الجامعية التوثيق الالكتروني والميكروفيلم



# جامعة عين شمس

التوثيق الإلكتروني والميكرو فيلم

## قسم

نقسم بالله العظيم أن المادة التي تم توثيقها وتسجيلها  
علي هذه الأقراص المدمجة قد أعدت دون أية تغييرات



## يجب أن

تحفظ هذه الأقراص المدمجة بعيدا عن الغبار





# SPEEDING UP EXPONENTIATION ALGORITHMS FOR PUBLIC-KEY CRYPTOSYSTEMS

Presented By  
Khaled AbdElrafea Abdelbari  
B.Sc. in Computer Science, Ain Shams university

Supervised By  
*Assoc. Prof.* Sameh Samy Daoud  
*Assoc. Prof.* Hatem Mohamed Bahig

A THESIS SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS  
FOR THE MASTER OF SCIENCE DEGREE  
IN COMPUTER SCIENCE  
TO  
DEPARTMENT OF MATHEMATICS  
FACULTY OF SCIENCE  
AIN SHAMS UNIVERSITY  
CAIRO, EGYPT  
2021



# AIN SHAMS UNIVERSITY

Author: **Khaled AbdElrafea AbdElbari**  
**B.Sc. in Computer Science, Ain**  
**Shams university**

Title: **Speeding Up Exponentiation**  
**Algorithms for Public-Key**  
**Cryptosystems**

Division: **Computer Science**

Department: **Department of Mathematics**

Faculty: **Faculty of Science**

Degree: **M.Sc.** Year: **2021**

# Contents

List of Tables	vii
List of Figures	ix
List of Algorithms	xi
List of Abbreviations	xii
Acknowledgements	xiii
Abstract	xiv
Summary	xv
<b>1 Introduction to Exponentiation Algorithms</b>	<b>1</b>
<b>2 Addition Chains</b>	<b>4</b>
2.1 Basic Definitions and Facts . . . . .	4
2.2 Generalization of ACs . . . . .	8
2.2.1 Addition Sequence . . . . .	8
2.2.2 Vectorial $AC$ . . . . .	9
2.2.3 Addition-Subtraction Chain . . . . .	9
2.2.4 Addition-Multiplication Chain . . . . .	10
2.3 Special Types of ACs . . . . .	10
2.3.1 Lucas-AC . . . . .	11
2.3.2 Euclidean-AC . . . . .	11
2.3.3 $\ell^0$ -AC . . . . .	12
2.4 ACs Representation . . . . .	12
2.4.1 Using Elaborating Method . . . . .	12

2.4.2	Using Directed Acyclic Graphs . . . . .	12
2.5	ACs Applications . . . . .	13
2.5.1	Encryption and Decryption . . . . .	13
2.5.2	Decision Tree Problem . . . . .	13
2.5.3	The Smallest Grammar Problem . . . . .	14
<b>3</b>	<b>Generating Addition Chains</b>	<b>15</b>
3.1	Generating Short ACs . . . . .	15
3.1.1	Binary AC Algorithm . . . . .	15
3.1.2	Window AC Algorithm . . . . .	19
3.2	Generating Shortest ACs . . . . .	21
3.2.1	Branch and Bound Algorithm . . . . .	21
3.2.2	B&B-DFS Shortest <i>AC</i> Algorithm . . . . .	37
3.2.3	B&B-BFS Shortest <i>AC</i> Algorithm . . . . .	42
3.2.4	MultiCore Shortest <i>AC</i> Algorithm . . . . .	44
<b>4</b>	<b>Introduction to GPU and CUDA</b>	<b>46</b>
4.1	Introduction . . . . .	46
4.2	Definitions . . . . .	47
4.3	NVIDIA GPU Architecture . . . . .	49
4.4	CUDA Programming Model . . . . .	51
4.4.1	CUDA Kernels . . . . .	54
4.4.2	Thread Organization . . . . .	56
4.4.3	CUDA Memory Hierarchy . . . . .	57
<b>5</b>	<b>Improving Generation of Shortest Addition Chains using GPUs</b>	<b>59</b>
5.1	A GPU-Based Shortest Addition Chain Algorithm . . . . .	59
5.1.1	Outline of the Proposed Algorithm . . . . .	59
5.1.2	The Proposed Algorithm . . . . .	61
5.1.2.1	Algorithm TTree . . . . .	62
5.1.2.2	Algorithm MTree . . . . .	63
5.1.2.3	Algorithm BTree . . . . .	66
5.2	Experimental Results . . . . .	68
5.2.1	Platform Specifications and Data Set . . . . .	68
5.2.2	Calculating the Depth $d$ for <b>TTree</b> . . . . .	68

5.2.3	Determining the Threshold <i>MinThread</i> and the Block Dimension for the Kernel . . . . .	70
5.2.4	Performance of <b>TMBTree</b> Algorithm . . . . .	73
5.2.5	Generating All Shortest Addition Chains . . . . .	75
5.2.6	Numerical Examples . . . . .	76
5.2.7	Shortest Star Addition Chains . . . . .	76
<b>6</b>	<b>Conclusion and Future Work</b>	<b>79</b>
	<b>Bibliography</b>	<b>80</b>

# List of Tables

3.1	Binary <i>AC</i> algorithm execution steps for $e = 55 = (110111)$ . . . . . .	17
3.2	right-left binary <i>AC</i> algorithm execution steps for $e = 55 = (110111)$ . . . . . .	18
3.3	Selecting bounding sequences (vertical and slant) of length $lb$ for $e = 2^t m$ , $t \geq 0$ , and $m$ is odd. . . . .	25
3.4	The averages of the maximum lengths of <i>DStack</i> and the total number of generated children with/without using the bounding sequences . . . . .	40
3.5	Speedup of Algorithm 3.6 using different numbers of cores [7] . . . . .	44
5.1	Average execution time in seconds, and the ratio for calculating the depth $d$ compared to <b>TMBTree</b> and sequential (Algorithm 3.4) . . . . .	70
5.2	Average number of kernels and threads using/without using the threshold $MinThread = 4096$ . . . . .	72
5.3	Average execution time in seconds for different values of <i>MinThread</i> (minimum number of elements in <i>CurBreadthLevel</i> ) and block sizes for generating a shortest <i>AC</i> for 200 random integers in the range $[2^{15}, 2^{22}]$ , i.e., 16 – 22 bits using the GPU (GTX 770). . . . .	73
5.4	Average of execution time in seconds for sequential and parallel ( <b>TMBTree</b> ) implementations to generate one shortest <i>AC</i> . . . . .	74
5.5	Obtained speedups using <b>TMBTree</b> to generate one shortest <i>AC</i> . . . . .	75

5.6	Comparison between <b>TMBTree</b> and the sequential version to generate all shortest <i>ACs</i> . The average execution time is in minutes . . . . .	75
5.7	Obtained speedups using <b>TMBTree</b> to generate all shortest <i>ACs</i> . . . . .	76
5.8	Execution times for some numbers . . . . .	76
5.9	Execution times and number of elements in the search tree for some numbers of the same bit size . . . . .	77
5.10	Average of execution time in seconds for Algorithm 3.4 and <b>TMBTree</b> implementations to generate one shortest star <i>AC</i> . . . . .	78

# List of Figures

2.1	DAG for $AC(55)$ . . . . .	13
3.1	Search tree up to level 4 . . . . .	22
3.2	Search tree of shortest $ACs(7)$ . . . . .	23
3.3	Search tree of shortest $ACs(17)$ . . . . .	24
3.4	Search tree of shortest $ACs(23)$ for the branch $\{1, 2, 4\}$ .	28
3.5	Search tree of shortest $ACs(23)$ using Eq. (3.1) as vertical bounding sequence . . . . .	29
3.6	Search tree of shortest $ACs(23)$ using Eq. (3.2) as vertical bounding sequence . . . . .	30
3.7	Search tree of shortest $ACs(23)$ using Eq. (3.3) as vertical bounding sequence . . . . .	31
3.8	Search tree of shortest $ACs(23)$ using Eq. (3.1) as slant bounding sequence . . . . .	32
3.9	Search tree of shortest $ACs(23)$ using Eq. (3.2) as slant bounding sequence . . . . .	33
3.10	Search tree of shortest $ACs(23)$ using Eq. (3.1) as vertical and slant bounding sequences . . . . .	34
3.11	Search tree of shortest $ACs(23)$ using Eq. (3.2) as vertical and slant bounding sequences . . . . .	35
3.12	Search tree of shortest $ACs(23)$ using Eq. (3.3) as vertical bounding sequence and Eq. (3.2) as slant bounding sequence	36
3.13	Average maximum DStack lengths with/without using the bounding sequences . . . . .	40
3.14	Logarithm of the average number of generated nodes (elements) in the search tree with/without using the bounding sequences . . . . .	41

4.1	Schematic of NVIDIA GPU Architecture [40] . . . . .	50
4.2	GPU vs CPU [40] . . . . .	51
4.3	CUDA program execution [40] . . . . .	52
4.4	Automatic CUDA Scalability [40] . . . . .	53
4.5	CUDA memories hierarchy [40] . . . . .	57
5.1	General idea of TMBTree . . . . .	60
5.2	Execution time of <b>TMBTree</b> using different GPUs . . . . .	74

# List of Algorithms

1.1	AC Exponentiation . . . . .	3
3.1	Binary AC . . . . .	16
3.2	Right-Left Binary AC . . . . .	18
3.3	Window AC . . . . .	19
3.4	B&B-DFS-Shortest-AC . . . . .	39
3.5	B&B-BFS-Shortest-AC . . . . .	43
3.6	Multicore-Shortest-AC [7] . . . . .	45
5.1	TMBTree . . . . .	61
5.2	TTree . . . . .	64
5.3	MTree . . . . .	66
5.4	BTree . . . . .	67
5.5	GetDepthLevel . . . . .	71

# List of Abbreviations

We use the following standard notations with a short explanation.

<b>Notation</b>	<b>Explanation</b>
$AC(e)$	Addition chain of an integer $e$
B&B	Branch and Bound
BFS	Breadth First Search
CUDA	Compute Unified Device Architecture
DFS	Depth First Search
GPUs	Graphics Processing Units
$Z_m$	Set of integers $\{0,1,\dots,m-1\}$ modulo a positive integer $m$