GENERATION OF ADDITION CHAINS USING EVOLUTIONARY ALGORITHMS FOR OPTIMIZING THE TIME IN MOBILE DEVICES

Thesis submitted to the Bharathidasan University, Tiruchirappalli in partial fulfilment of the requirements for the award of degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

Submitted By

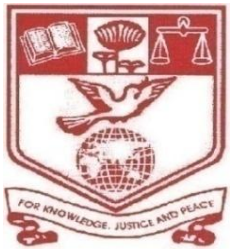
A. MULLAI

Ref. No: 16634/Ph.D.-K3/Computer Science/Part Time/July 2014

Under the Guidance of

Dr. K. MANI

Associate Professor & Research Advisor



PG AND RESEARCH DEPARTMENT OF COMPUTER SCIENCE

NEHRU MEMORIAL COLLEGE (AUTONOMOUS)

(Nationally Accredited with 'A+' Grade by NAAC)

PUTHANAMPATTI , TIRUCHIRAPPALLI - 621 007

TAMIL NADU, INDIA.

MARCH 2022

Curiginal

Document Information

Analyzed document	Mullai_Doc.pdf (D131761366)
Submitted	2022-03-28T09:57:00.0000000
Submitted by	Srinivasa ragavan S
Submitter email	bdulib@gmail.com
Similarity	0%
Analysis address	bdulib.bdu@analysis.urkund.com

Sources included in the report

Dr. K. Mani
Associate Professor
PG & Research Department of Computer Science
Nehru Memorial College (Autonomous)
Nationally Accredited with 'A+' Grade by NAAC
(Affiliated to Bharathidasan University by UGC)
Puthanampatti - 621 007, Tiruchirappalli – District
Tamil Nadu, India

Certificate

This is to certify that the thesis entitled **"Generation of Addition Chains Using Evolutionary Algorithms for Optimizing the Time in Mobile Devices"**, submitted by **Mrs.A. Mullai**, a Research Scholar, PG & Research Department of Computer Science, Nehru Memorial College (Autonomous), Puthanampatti - 621 007, Tiruchirappalli, Tamil Nadu, India for the award of the degree of **Doctor of Philosophy in Computer Science**, is a record of original work carried out by her under my supervision and guidance. The Thesis has fulfilled all requirements as per the regulations of the University and in my opinion the thesis reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Date:

Place: Puthanampatti

Dr. K.Mani

Research Supervisor

A. Mullai
Research Scholar
PG & Research Department of Computer Science
Nehru Memorial College (Autonomous)
Nationally Accredited with 'A+' Grade by NAAC
(Affiliated to Bharathidasan University by UGC)
Puthanampatti - 621 007, Tiruchirappalli – District
Tamil Nadu, India

Declaration

I hereby declare that the work embodied in this thesis entitled **"Generation of Addition Chains Using Evolutionary Algorithms for Optimizing the Time in Mobile Devices"**, is a research work done by me under the supervision and guidance of **Dr. K. Mani**, Associate Professor, PG & Research Department of Computer Science, Nehru Memorial College (Autonomous), Puthanampatti - 621 007, Tiruchirappalli, Tamil Nadu, India. The thesis or any part there of has not formed the basis for the award of any Degree, Diploma, Fellowship, or any other similar titles.

Date:

Place: Puthanampatti

(A. Mullai)

ACKNOWLEDGEMENT

-God is great-

First and foremost, I thank ***Almighty God*** for being with me in all my happiness and sorrows to complete my research work in time. Nothing can happen without spiritual prayer.

I would like to extend my sincere and deep sense of gratitude to my Research Advisor **Dr. *K. Mani***, Associate Professor in Computer Science, Nehru Memorial College (Autonomous), Puthanampatti for his invaluable suggestions, patience, motivation, enthusiasm and immense mathematical knowledge, meticulous editing, support and tutelage during the course of my Ph.D programme. His invaluable guidance helped me in all the time of publishing the research papers and writing the thesis in an efficient and effective manner.

I express my deep sense of gratitude to the college management especially, ***Mr. Pon Balasubramanian,*** President and ***Mr. Pon Ravichandran,*** Secretary, Nehru Memorial College (Autonomous), Puthanampatti for providing opportunity and support to do my research in this esteemed institution.

I am extremely thankful to the Principal, ***Dr. A. R. Pon Periyasamy***, Nehru Memorial College (Autonomous), Puthanampatti for providing the facilities to carry out my research work successfully.

My special thanks to my Doctoral Committee Members, ***Dr. D.I. George Amalarethinam***, Bursar, Director (MCA) & Associate Professor in Computer Science, Jamal Mohamed College (Autonomous), Tiruchirappalli and ***Dr. E. George Dharma Prakash Raj***, Associate Professor, School of Computer Science, Engineering and Applications, Bharathidasan University, Tiruchirappalli for their valuable suggestions to enhance my research work effectively. My heartfelt thanks to ***Dr. M. Muralidharan***, Associate Professor and Head, Department of Computer Science, Nehru Memorial College (Autonomous), Puthanampatti.

I would like to thank our Management trustee and **Secretary Sri. R. Panchapakesan** and **Co-ordinator Smt. Vasantha Panchapakesan**, Seethalakshmi Ramaswami College (Autonomous), Tiruchirappalli for their blessings and support to do my research. I extend my sincere thanks to our **Executive Director Sri. Ramani**

Panchapakesan, and **Dr. Kannan Panchapakesan, Director-Academics**, Seethalakshmi Ramaswami College (Autonomous), Tiruchirappalli for their constant support towards the completion of my research work.

I would like to thank our **Principal Dr. M. Vasuki**, for the moral support and guidance towards the completion of my research work.

I extend my sincere thanks to our Head **Dr.K.S.Rathnamala** and other staff members **Dr.R.Jamuna**, **Dr.S.Lakshmi Prabha**, **Ms.K.Mahalakshmi**, **Ms.V.Gayathri and Ms.A.M.Aarthi** for their encouragements and support for the completion of my research work. I extend my sincere thanks to the faculty members of computer Science department and other department, all teaching and non-teaching staff members for their encouragement and support for me.

I should appreciate and thank Mr.***T. L. Kannan*** for his timely support towards my research work. I thank all my co-research scholars, ***Dr. Mohana Krishnan, Dr. Devi, Dr. Kalpana, Dr. Elavarasan, Dr. Mahendran, Dr. Viswambari, Dr. A. Barakath Begam, Mrs. A. Akila*** *and* ***Mr. Prasath Sivasubramaniyan,*** for their timely help.

Finally I owe my deepest gratitude to my Parents, ***Rtd. Prof. C. N. Arul Prakasam***, ***A. Bhagavathi,*** Father-in-law ***Tashildar(Late) K. Muthuswamy*** mother in-law ***Retd. Headmistress S. Soundranayaki,*** my Husband ***Prof. Dr. M. Sundar*** and my family members for their blessings, understanding, affection and support for me in everything. I appreciate my lovable Son ***Shri***. ***M. S. Shyam Sundar (B.Tech)*** for understanding my situation that enabled me to complete my thesis successfully. I wish to extend my special thanks to my brother in-law, sister in-laws, brothers and sisters for their encouragement and motivation towards my research. It's their blessings and prayers that enabled me to complete this work.

The living goals on the earth are parents and teachers. I humbly submit this thesis to them. It is the result of their care and up-bringing. Above all, I thank God, the Almighty for bestowing me with abundant grace especially when I waded through great obstacles in my life.

A.MULLAI

ABSTRACT

The usage and applications of mobile devices are increasing exponentially day by day. The applications of smart mobile phones are also increasing which lead to many security issues. The security features can be taken and applied on these devices. Mobile devices deal with heterogeneity of networks and also in ubiquitous intelligent environment with embedded computers everywhere and reliable services to the user in an easy way. Mobile computing will enable the transmission of voice, video and data between human and the computer. It always helps to stay connected to the world with a wide range of users through the internet. The devices are primarily designed to make for communication purpose but now people started to do all sort of works through these devices. They engage people with entertainment, education, teaching, money transaction, communication, games and all social media apps. Moreover, the devices have limited battery power and storage. Even though, they have more offerings to the user, lot of challenges like disconnection, low/high bandwidth variability, low power and resources, security risks, wide variety of devices with different capabilities and to fit more functionality into single, smaller devices.

Thus, it is necessary to optimize their battery power and fixed limited space for storage which makes serious issues, challenges and threatening from hackers are analyzed. Security threats also arise while transferring sensitive informations through mobility as well as wireless devices like mobile devices. All sensitive informations have been transmitted with high speed as much as possible through these handheld mobile devices. Hence, there are five parameters have been taken in this work viz., encryption time, decryption time, encryption power, decryption power and security.

One essential aspect for secure communications is using cryptography. Cryptography is the most indispensable tool for keeping information in secure manner in any computing system and it has been taken into mobile computing systems/devices too. Even though, many symmetric-key algorithms viz., AES, SERPENT and TWOFISH are used for providing security in mobile devices, but they are not providing that much security because they use only basic operations like shifting the bits, initial permutation, mix column transformation etc., and hence, the public-key algorithms like Rivest Shamir Adleman (RSA) and Elliptic Curve Cryptography (ECC) have been taken in this work because they involve some complicated mathematics which provide more security than symmetric-key encryption algorithms.

RSA is a procedure of computational simplicity whereas ECC provides greater security. Large prime numbers are used as security keys in these methods. As the key size taken in RSA and prime number used in ECC are very large, they may take more time for encryption and decryption. It is noted that if a cryptographic algorithm takes more time in performing operational time (where the operational time includes both encryption and decryption) which causes customer's impatience and dissatisfaction. Thus, to decrease the operational time in RSA and ECC, Addition Chain (AC) is incorporated in performing $x^e \bmod n$ of RSA and $k[P]$ of ECC. There are many algorithms exist in literature to generate the AC. But, the bioinspired based algorithms viz., Particle Swarm Optimization (PSO), Simplified Swarm Optimization (SSO) and Bacteria Foraging Optimization (BFO) algorithms are taken in this work to generate the ACs for an integer n and they are termed as AC-PSO, AS-SSO and AC-BFO respectively.

PSO is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. Dr. Eberhart and Dr. Kennedy proposed PSO in 1995, on the basis of flocking birds' social activity and fish schooling. It is a meta-heuristic algorithm. In PSO, all the birds do not know where food is but they know how they move in each iteration. In PSO, each member of the population is called particle and the population is called swarm.

All particles are modified on the basis of the two best values after each iteration. The first best value called pBest is already obtained by a particle. The second-best value called $gBest$ is the best value achieved for the fitness function tracked by the particle swarm optimizer by the general population. Since each parameter tries to modify the position by using the information viz., (i) the current position, (ii) the current velocity (iii) the distance between the current position and $pbest$ (iv) the distance between the current position and gbest. Based on these, the new velocity and new positions are calculated.

In this work, particle represents the AC, velocity represents the number to be added to the current number (position) x_i so that the next number x_{i+1} is obtained. Further, fitness function is taken as length of AC denoted as l(n). In this work, $c_1=c_2=0.7$ where 0.7 is a uniform random number. Similarly, other random numbers r_1, r_2 are taken from RAND corporation table. When RSA and ECC are considered, the key is taken very large, and AC of the key is generated according to the proposed AC- PSO.
SSO is a population-based, evolutionary, stochastic optimization technique in soft computing and it was originally designed by Yeh. It has some advantages, such as fast convergence rate, few parameters, and easy implementation. It has simple procedures and more powerful global searching, prevents from trapping local optimal procedures.

The major difference among SSO and other soft computing algorithms are their update mechanism (UM). However, the UM of SSO is based on NP-hard problem. In SSO, each Chain Particle (CP) represents the AC. The search spaces for the elements are often restricted to simplifying the method of optimization. The first CP is 1 since all ACs should start with 1. The second CP is 2 with a value of 1 doubled. There are no optimization processes involving the first two elements. The third CP elements are either $3\ (2\ +\ 1)$ or a $4\ (2x2\ )$ and $5, 6$ or 8 may be the fourth part. After completing all epochs, particle outputs are optimized for SSO particles with CP elements. Bacteria foraging is one of the optimization and evolutionary algorithms. It was proposed by Kevin M. Passino in 2000 and it has been widely accepted as a new nature-inspired optimization algorithm. It is inspired by the social foraging behavior of *Escherichia coli* .

The foraging strategy of E.coli is achieved by four processes viz., chemotaxis, swarming, reproduction and dispersal. Chemotaxis is a process which simulates the movement of E.coli cell through swimming and tumbling via flagella. Movement of E.coli bacterium can be performed in two ways viz., (i) swim for a period of time in the same direction or it may tumble (ii) alternate between swim and tumble for the entire lifetime. In swimming process, a group of E.coli cells arrange themselves in a travelling ring by moving up the nutrient gradient when placed amidst a semisolid matrix with a single nutrient chemo-effecter. The healthy bacteria asexually split into two bacteria, which are then placed in the same location while the least healthy bacteria eventually die in reproduction process. In elimination and dispersal process, gradual or sudden changes in the local environment i.e., significant local rise of temperature or due to unavoidable events all the bacteria in a region are killed or a group is dispersed into new location.

In the proposed AC-BFO methodology, the concept of BFO is used to generate the optimum length AC for an integer . In this optimization, each bacterium represents AC. The cost or fitness function is computed with minimum length approach based on the nutrient concentration of the immediate environment of the bacterium searching for numbers in AC. Swarming step is not considered for the generation of AC in this method.

All the proposed methodologies are implemented in VC++ with Android and Windows emulators. A good cryptographic algorithm should be capable of processing with reasonable power consumption without compromising the security strength. The proposed AC-PSO, AC-SSO and AC-BFO are incorporated into RSA and ECC cryptosystems which are widely used in mobile devices with different file sizes of plaintext viz.,1MB, 2MB, 4MB, 8MB and 16 MB and the parameters like encryption time, decryption time, power consumed for encryption and decryption are computed. Similarly, security levels are measured using All Block Cipher (ABC) Universal Hackman tool. As the mobile devices are battery powered devices. They provide greater mobility and achieving higher security levels with lesser time and power consumption are the ultimate aim of a ideal cryptography procedure. Different size files with parameters are recorded in tables.

It is noted that to transmit any file in secure manner, it should be encrypted. Based on two different OS based emulators, the results show that the time taken for encryption, decryption, encryption power and decryption power of RSA and ECC with android OS take more time than RSA using window OS. It is also evident from the experimental results that the time taken for encryption, decryption, encryption power and decryption power using AC-PSO-RSA and AC-PSO-ECC with android OS

emulator takes more time than with window OS emulator. Generally, ECC taken more time than RSA. Hence, it is proved that AC-BFO-RSA is taking less time than AC-SSO-ECC. It is known fact that ECC provides more security than RSA. It is also proved in PSO, SSO and BFO based AC, when they are incorporated into RSA and ECC combinations. As there is directly proportional relation between time and battery power consumption, it is proved experimentally too. When security is concerned, the AC-BFO-ECC is recommended.

In order to support the work, the researcher has published four papers. Among them, one paper is published in Springer Nature.

LIST OF PUBLICATIONS / COMMUNICATIONS

International Journals

[1]. Dr.K.Mani and A.Mullai, "A Survey on the Security Features of Cryptographic Techniques in Mobile Devuces", *International Journal of Innovative Research in Computer and Communication Engineering*, vol.4, issue 2, February 2016.

[2]. Dr.K.Mani and A.Mullai, "Optimizing the Run Time in Mobile Devices", World Congress on Computing and Communication Technologies (WCCCT)", 2-4, IEEE *Xplore*, DOI: 10.1109/WCCCT, February 2017.

[3]. A.Mullai, Dr.K.Mani, "Enhancing the Security in RSA and elliptic curve cryptography based on addition chain using simplified Swarm Optimization and Particle Swarm Optimization for mobile devices", *International Journal of Information technology*, Springer, vol.13, pp.55-564, 2021. https:// doi.org/10.1007/s41870-019-00413-8.**[UGC-CARE-SCOPUS INDEXED]**

[4]. Dr.K.Mani and A.Mullai, "Generation of Addition Chain using Bacteria Foraging Optimization Algorithm", *International Journal of Engineering Trends and Technology*, vol. 69, issue 2, pp.32-38, DOI: 10.14445/22315381 /ijett-v69i2p205. **[UGC-CARE-SCOPUS INDEXED]**

International Conference

[1]. Dr.K.Mani and A.Mullai, "Optimizing the Run Time in Mobile Devices", World Congress on Computing and Communication Technologies (WCCCT)", 2-4, IEEE *Xplore*, DOI: 10.1109/WCCCT.2016.23, February 2017.

CONTENTS

LIST OF FIGURES

... LIST OF FIGURES

... LIST OF FIGURES

... LIST OF FIGURES

... LIST OF FIGURES

LIST OF TABLES

… LIST OF TABLES

… LIST OF TABLES

… LIST OF TABLES

LIST OF ALGORITHMS / PSEUDOCODES

LIST OF ABBREVIATIONS

ABC	All Block Ciphers Universal Hackman tool
ACO	Ant Colony Optimization
AES	Advanced Encryption Standard
AI	Artificial Intelligence
AIS	Artificial Immune System
APAU	Analysis and Prediction of Application Usage
ASCII	American Standard Code for Information Interchange
BFA	Brute Force Attack
BFO	Bacteria Foraging Optimization
BFOA	Bacteria Foraging Optimization Algorithm
BSA	Bird Swarm Algorithm
CBC	Cipher Block Chaining
CFB	Cipher Feed Back
DES	Data Encryption Standard
DLP	Discrete Logarithm Problem
DoS	Denial of Service
EAs	Evolutionary Algorithms
EC	Elliptic Curve
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDLP	Elliptic Curve Discrete Logarithm Problem
ECDSA	Elliptic Curve Digital Signature Algorithm

... LIST OF ABBREVIATIONS

ELS	Extending Local Search
FND	First Node to Die
GA	Genetic Algorithm
GBMAC	Graph Based Minimal Addition Chain
GUI	Graphical User Interface
HTTPS	Hyper Text Transfer Protocol Secure
IBE	Identity Based Encryption
ICT	Information Communication and Technology
IDEA	International Data Encryption Algorithm
IFP	Integer Factorization Problem
ITA	Itoh-Tsujii algorithm
ITU - T	International Telecommunication Union-T
KPA	Known Plaintext Attack
LM	Loss Minimization
LPC	Lattice Path Cryptosystem
LR	Lattice Reduction
LWE	Learning With Errors
MANET	Mobile Ad-hoc NETworks
MD5	Message Digest Algorithm 5
MITA	Modified ITA algorithm
MOS	Mobile Operating Systems
MPSO	Mutation based PSO
MWSNs	Mobile Wireless Sensor Network

... LIST OF ABBREVIATIONS

NIST	National Standards of Institute and Technology
OAC	Optimal Addition Chain
O-ECC	Optimized ECC
OS	Operating Systems
P2P	Peer to Peer
PKC	Public Key Cryptography
PKCS	Public-Key Cryptography Standards
PKI	Public-key Infrastructure
PSO	Particle Swarm Optimization
PSOFIM	PSO with Full Information and Mutation Operator
QoS	Quality of Service
RC4	Rivest Cipher 4
RLE	Run-Length Encoding
RNG	Random Number Generator
RRAP	Reliability Redundancy Allocation Problems
RSA	Ron Rivest, Adi Shamir, and Leonard Adleman
SMS	Short Message Service
SSH	Secure Socket Layer
SSO	Simplified Swarm Optimization
SSO-MS	SSO with Modular Search
SSP	Sum of Subsets Problem
UI	User Interface
UM	Update Mechanism

... LIST OF ABBREVIATIONS

VUI	Voice User Interface
Wi-Fi	Wireless Fidelity
WSN	Wireless Sensor Networks
WWW	World Wide Web

LIST OF SYMBOLS

Symbol	Meaning
F_p	finite field
(k_e, k_d)	Public key and Private key pair
ED_n	Number of elimination-dispersal events
Gb, g_{best},	Global-best value
K_s	Keystream
P_{ed}	Elimination-dispersal probability
Q_R(p)	Quadratic Residue
RP_n	Number of reproduction steps
c_1, c_2	Learning factors / Accelerating factors
$x_{i,j}^t$	j^{th} variable of X_i^t
γ_1,γ_2	Random numbers between 0 to 1
$[k]P$	Scalar point multiplication
{x, y}	Pairs of integer coordinates
||	Concatenation
∀	For All
∃	There Exists
∈	Element of
∉	Not an element of
≡	Identical to
⊂	Subset of
A	Android emulator
AC	Addition Chain

... LIST OF SYMBOLS

Symbol	Meaning
AC-BFO- ECC	AC based on BFO used in ECC
AC-BSO-RSA	AC based on BFO used in RSA
AC-BSO-RSA -A	AC based on BFO used in RSA(Android Emulator)
AC-BSO-RSA -W	AC based on BFO used in RSA(Windows Emulator)
AC-ECC	AC used in ECC
AC-PSO-ECC	AC based on PSO used in ECC
AC-PSO-RSA	AC based on PSO used in RSA
AC-PSO-RSA-A	AC based on PSO used in RSA (Android Emulator)
AC-PSO-RSA-W	AC based on PSO used in RSA (Windows Emulator)
AC-RSA	AC used in RSA
AC-SSO- ECC	AC based on SSO used in ECC
AC-SSO-RSA	AC based on SSO used in RSA
AC-SSO-RSA-A	AC based on SSO used in RSA (Android Emulator)
AC-SSO-RSA-W	AC based on SSO used in RSA (Windows Emulator)
b_1	bacteria
C or Y	Ciphertext
CP	Chain Particle
D(C)	Decryption of Ciphertext
$D_K(C)$	Decrypting the Ciphertext with key K
DP	Decryption Power
DT	Decryption Time
e	Exponent / Random integer

... LIST OF SYMBOLS

Symbol	Meaning
$E_K(M)$	Encryption of plaintext with key K
EP	Encryption Power
EQF	Equivalent Quadratic Form
ET	Encryption Time
G	Primitive Root
GBAPAC	Graph Based All Possible AC
gBest	Global Best
GBMAC	Graph Based Minimum number of AC
g^n	Group element multiplied by itself n times
IP	Intermediate Plaintext
l(n)	Length of AC
lBest	Local best
M or P or X	Plaintext
m_i	Plaintext character
$\mathbb{N}$	Natural numbers
pBest	Particle Current Best value
PKC	Public-Key Cryptography
$Q_{NR}(p)$	Quadratic- Non Residue with Prime P
$\mathbb{R}$	Real numbers
Rand ()	Random number
RAs	Repeated Additions
RMs	Repeated Multiplications

... LIST OF SYMBOLS

Symbol	Meaning
SE	Security Level
SE-AC-BFO-RSA-A	Security level produced by AC based on BFO in RSA (Android Emulator)
SE-AC-BFO-RSA-W	Security level produced by AC based on BFO in RSA (Windows Emulator)
SE-AC-PSO-RSA-A	Security level produced by AC based on PSO in RSA (Android Emulator)
SE-AC-PSO-RSA-W	Security level produced by AC based on PSO in RSA (Windows Emulator)
SE-AC-SSO-RSA-A	Security level produced by AC based on SSO in RSA (Android Emulator)
SE-AC-SSO-RSA-W	Security level produced by AC based on SSO in RSA (Windows Emulator)
SKC	Symmetric-Key / Single-Key / Private-key Cryptography
V	Velocity
W	Windows emulator
$\mathbb{Z}$ or Z	Set of integers
ω	Inertia Weight
BS	Bit Stream
D	Decryption / Deciphering
E	Encryption / Enciphering
K	Key
O	Point on Infinity
S	Total number of bacterium in the population
S_w	The swimming length
d	Dimension of the search space. Here, $d = 1$
g	Finite group G

CHAPTER – I

INTRODUCTION

1.1 Background

As the world becomes more connected, the demand for sending the sensitive data like credit and debit card numbers, money transaction, sharing the message in military etc., through the communication channel is increasing exponentially day by day. It must be protected from the third party called adversary so that he/she could not understand the meaning of such message. In order to protect such sensitive data, the indispensable tool cryptography is used [1]. Cryptography is originated from the Greek word *kryptos*, meaning "hidden," and the word *graphein*, means "to write" which is the process of converting plaintext to ciphertext and vice-versa. It is a process of transmitting and storing data in a specific form so that the authenticated person can only read and process it. It is not only used to protect data from theft or alteration, but it can also provide many security services viz., data integrity, confidentiality, non-repudiation and authentication as defined in ITU-T (International Telecommunication Union-T) X.800 recommendation [2].

The service confidentiality refers to the protection of transmitted data from various attacks particularly passive attacks where passive attack attempts to learn or make use of information from the system but it does affect the system resources. The authentication service is concerned with assuring that a communication system is authentic, i.e., it is the process of user's identity. There are two types of authentication viz., peer entity authentication, data origin authentication. Peer entity authentication is mainly used in association with a logical connection to achieve the identity of the

entities connected. Data origin authentication is a connectionless transfer which provides assurance that the source of received data is as claimed [3].

The service data integrity deals with a stream of messages, assures that messages are sent, with no duplication, insertion, modification, recording, or replays. This service also covers destruction of data. Non-repudiation is a service which provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication. It consists of two types: non-repudiation-origin which is a proof that the message was sent by the specified party and non-repudiation-destination which is a proof that the message was received by the specified party [4].

1.2 Terminologies Used in Cryptography

In cryptography, original or any readable message is called plaintext M $(or\ X)$, and coded or unreadable message is called ciphertext, $C(or\ Y)$. Encryption or enciphering E, is a process of applying mathematical function to convert M into C. Decryption or deciphering D, is the reverse process of encryption or restoring M from C. A key, in cryptography or cryptographic key (K) is string of bits used in cryptographic algorithms to transform M into C or vice versa. The process of E and D is called cryptography. Cryptographic system or cipher includes cryptography and key K. Cryptanalysis is a method to break the code, i.e., recovering M from C without knowing K. The area of cryptography and cryptanalysis together is called cryptology [5]. Mathematically, E and D are represented as $C = E_k(M)$ and $C = D_k(C)$ respectively. Further, $D\big(E(M)\big) = M\ and\ E\big(D(C)\big) = C$. The basic principle used in modern cryptography is Kerckhoffs's principle. It is mentioned as "A cryptographic

system should be secure if everything about the system, except the key, is public knowledge". The concept of cryptography is shown in fig. 1.1.

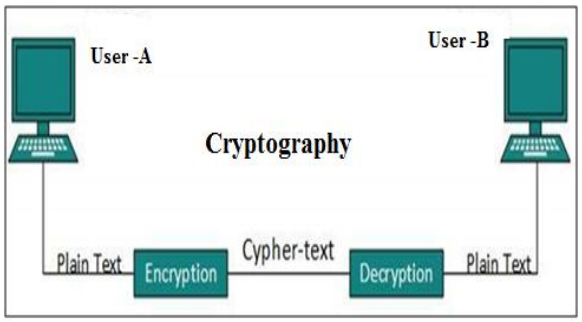


Fig. 1.1: Cryptography Block Diagram

1.3 Types of Cryptography

Cryptography is mainly classified into two types, viz., classical and modern cryptography. It is shown in fig. 1.2.

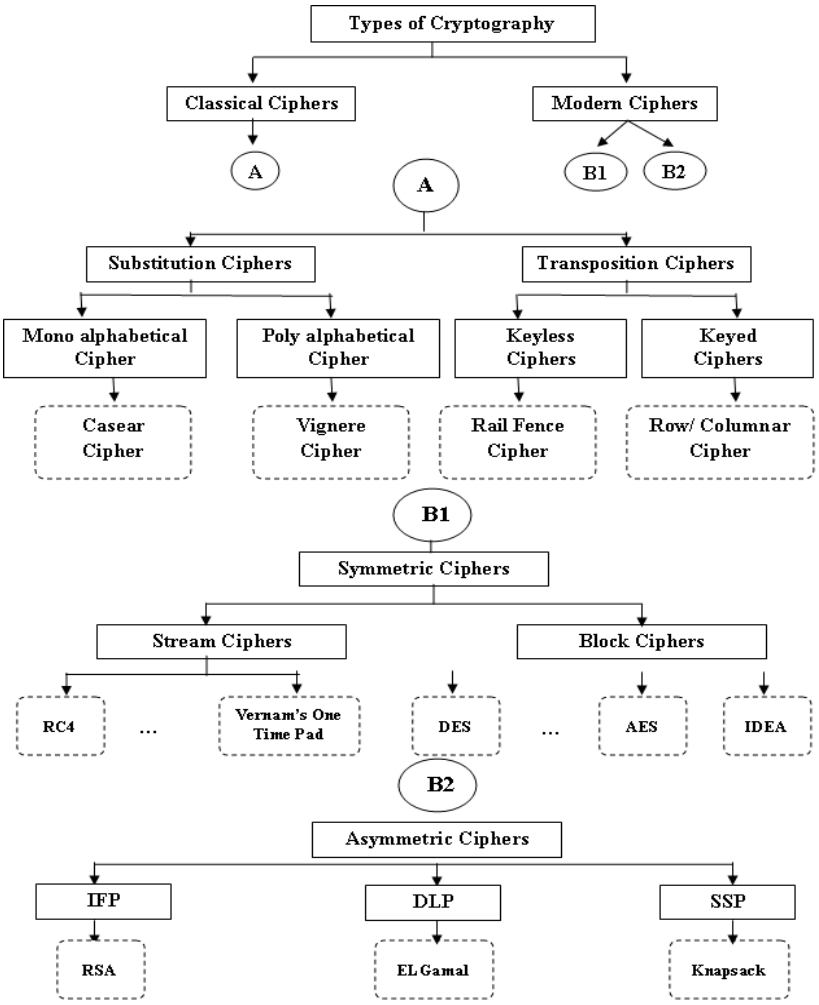


Fig. 1.2: Types of Cryptography

1.3.1 Classical Cryptography

In classical cryptography, no mathematical concepts are involved. The traditional characters, i.e., letters and digits are directly manipulated in this type. It is focused mainly on 'security through obscurity'. The parties involved in communication must know about the coding that were kept secret. It has two basic components substitution and transposition cipher. In substitution cipher, each $m_i \in M$ is replaced by other letter or symbol. If M is viewed as a sequence of bits, substitution involves replacing M bit patterns with C bit patterns. One of the earliest known and simplest substitution cipher was Ceaser cipher. It was invented by Julius Ceaser. In substitution cipher, each letter in M is shifted a certain number of places down the alphabet. A shift may be any value, and the general Julius Caeser cipher encryption algorithm is $c_i = E(m_i) = (m_i + k)\ mod\ p$, for each $c_i \in C$, $m_i \in M$ and $k, \in K$ and $k, p = 0,1, \dots ,25$. The decryption algorithm is $m_i = D(c_i) = (c_i - k)\ mod\ 2$ Other substitution cipher includes monoalphabetic cipher, playfair cipher, Hill cipher, polyalphabetic cipher etc. Transposition cipher is achieved by performing some kind of permutation on the plaintext letters, i.e., M remains same, but the order of characters is shuffled around. Simple columnar transposition cipher, German ADFGVX cipher etc., are some examples of transposition cipher [6][7]. In this cipher, if M has n characters, then the total number of possible ciphertexts produced using this method is $n!$.

1.3.2 Modern Cryptography

It is based on numerous ideas of mathematics such as number theory, computational-complexity theory and probability theory. It operates on binary bit sequences. In modern cryptography, secrecy is obtained through a secret-key which is used as the

seed for the algorithms. Absence of secret-key is the computational difficulty of these algorithms which make it impossible for an attacker to obtain the original information even if he/she knows the algorithm used for coding. Modern cryptography is divided into two types viz., Symmetric-Key Cryptography (SKC) and Public-Key Cryptography (PKC) [8][9].

- **Symmetric-Key Cryptography**

It is also called secret-key or single-key or private-key algorithm. It deals not only with E but also deals with authentication [10]. It requires that sender and receiver must share the same secret-key for performing both E and D. It is shown in fig. 1.3.

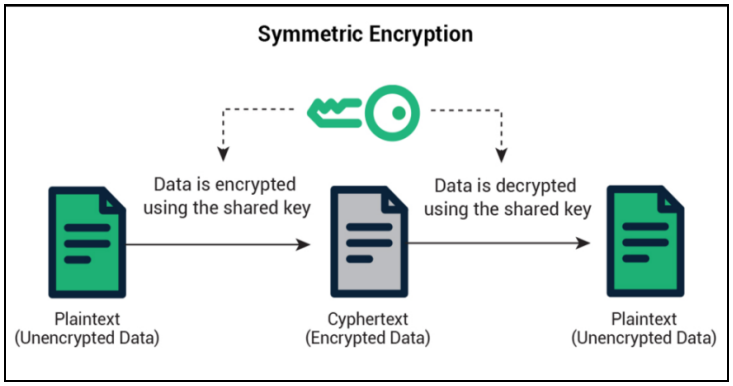


Fig. 1.3: Symmetric-key Cryptography

The security of SKC rests in the key; divulging the key means that anyone could encrypt and decrypt the messages. As long as the communication needs to remain secret, it must be remain secret. Even though, it is faster than public-key cryptography, the main problem with this algorithm is getting the sender and receiver to agree on the secret-key without anyone else finding out [11]. Mathematically, a symmetric key cryptosystem can be defined as the tuple (P, C, K, E, D) where P is the set of finitely many possible plaintexts, C is the set of finitely many possible ciphertexts and K represents the key space i.e., set of finitely many possible keys.

$\forall k \in K, \exists E_k \in E$ (encryption rule), $\exists D_k \in D$ (decryption rule) $E_k : P \rightarrow C$ and $D_k : C \rightarrow P$ are well defined functions such that $\forall p \in P, D_k\left(E_k(P)\right) = P$. SKC is further divided into stream ciphers and block ciphers. In stream cipher, one bit of character is encrypted at a time, i.e., it operates on smaller units of M, usually bits. Encryption is accomplished by a sequence of bits called a keystream (K_s) by combining the K_s with M, usually with the bitwise XOR operation.

Mathematically, stream cipher can be defined as $E^* : M^* \times K^* \rightarrow C^*, E^*(m,k) = c = c_1c_2c_3 \ldots$ Encrypts a stream $m = m_1m_2m_3 \ldots \in M^* \, of \, characters \; m_i \in M$ as a stream $c = c_1c_2c_3 \ldots \in C^* of$ ciphertexts $c_i \in C$ by using a key streams $k = k_1k_2k_3 \ldots \in k^*, k_i \in K$. One-Time Pad (OTP) is an example of stream cipher. In OTP, M=K=C={0, 1} and $E: \{0,1\} \times \{0,1\} \rightarrow \{0,1\}, (m,k) \rightarrow m \oplus k$. To encrypt a message $m = m_1 m_2 m_3 \ldots m_i \in \{0,1\}$, a key stream $k = k_1 k_2 k_3 \ldots k_i \in \{0,1\}$, is needed. Encryption and decryption are given by $E^*(m,k) = c = c_1 c_2 c_3 \ldots$, where $c_i = m_i \oplus k_i$ and $D^*(c,k) = c_1 c_2 c_3 \ldots$, where $m_i = c_i \oplus k_i$.

In block cipher encryption, group of bits or characters called block is encrypted at a time. It transforms fixed-length block of *M* data into block of *C* data of the same length. Mathematically, block cipher is a symmetric-key encryption scheme with $M = C = \{0,1\}^n$, n is called block length of the cipher. There are various block cipher modes exist viz., Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher FeedBack (CFB) and Output FeedBack (OFB) exist. Data Encryption Standard (DES), Advanced Encryption Standard (AES), Blowfish,RC4 (Rivest Cipher) etc., are some examples of block cipher symmetric-key encryption algorithms [12].

- **Public-Key Cryptography**

It is also called as asymmetric-key cryptography in which a pair of keys is involved known as public-key and private-key(k_e, k_d). Each public-key is visible and the corresponding private-key is kept secret. They are mathematically related but not identical. In PKC, each key performs a unique function unlike symmetric-key algorithms that rely on one key to both encrypt and decrypt. In PKC, for encryption and decryption public-key and private-key is used respectively [13]. Diffie-Hellman invented the concept of public-key in 1976. It is shown in fig. 1.4.

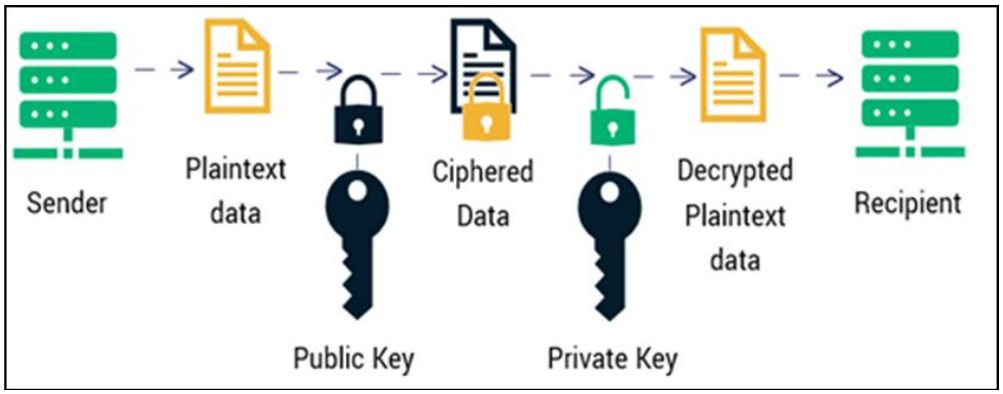


Fig. 1.4: Asymmetric-key or Public-key Cryptography

The main idea involved in PKC is that it is easy to derive k_e from k_d but it would be infeasible to derive k_d from k_e called trapdoor one-way function. RSA, ElGamal, McEllice, Rabin, Elliptic Curve Cryptography (ECC) etc., are some examples of PKC. It is further classified into three types viz., Integer Factorization Problem (IFP), Discrete Logarithm Problem (DLP) and Sum of Subsets Problem (SSP). IFP is defined as for a given positive integer n, compute its decomposition into prime numbers $n = p_i^{ei}$ (unique up to reordering). IFP is the act of splitting into an integers called factors which, when multiplied together, form the original integer. For example, for the integer

n = 27997833911221327870829467638722601621070446786955428537560
00992932612840010760934567104295536085606182235191095136578863
71059544820065767750985809557613579098730950144178863178946295
187237869221823983.

Its two prime factors are

(i) p = 35324619344027701212726049781984643686711974009762502364930346877612125367943200058547956528088349

(ii) q = 92586995447833303334708584148005968773797585736421996073433034145576787281815213538140930474185467

But, for the given n it is hard to find the factors p and q. The security of RSA algorithm depends on integer factorization. DLP is applied to mathematical structures called group. For a group element g and a number n, let g^n denotes the element obtained by multiplying g by itself n times, i.e., $g^n = g^* g^* \ldots^* g$. The DLP is defined as given an element g in a finite group G and another element $h \in G$, find an integer x such that $g^x = h$. For example $5^x \equiv 1 \bmod 19$ is 9. ElGamal encryption is based on DLP [14]. SSP is also called knapsack cryptosystem. It is an NP complete problem. It is defined as for the given positive integer weights $a = (a_1, \ldots, a_n)$ and $b = \sum_{i=3}^{n} a_i x_i \in z$ for the same bits $x_i \in \{0,1\}$, find $x = \{x_1, \ldots., x_n\}$. Merkle-Hellman Knapsack cryptosystem is based on SSP. Even though, many PKC exist in the literature, the two famous popular PKC algorithms RSA and ECC play a vital role in mobile devices and they are explained in the next subsections.

1.4 Rivest Shamir Adleman Algorithm

RSA is an asymmetric cryptography algorithm. Asymmetric works on two different keys i.e. public-key and private-key. It is based on the fact that it is very difficult to

factorize a large integer. The public-key has two numbers where one number is the multiplication of two large prime numbers while another number is private-key and it is derived from the same two prime numbers. Hence, anyone can factorize the large number, the private-key is compromised. Therefore, the encryption strength is totally depending on the key size and if double or triple the key size, the strength of encryption increases exponentially [15]. In RSA, the key size can be 2048 or 4096 bits.

1.5 Elliptic Curve Cryptography

ECC is a modern public-key encryption technique and it is based on the mathematical ECs. It is smaller, faster, and more efficient cryptographic keys, based on the algebraic structures of the ECs over finite fields and on the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP). It implements the major capabilities of asymmetric cryptosystems such as encryption, key exchange and signatures. This is considered as a natural modern successor of the RSA cryptosystem. Since, it uses smaller keys and signatures than RSA for the same level of security also provides very fast key generation, agreement and fast signatures [16].

This algorithm is used in the Secure Sockets Layer (SSL) standard for signing SSL certificates with ECDSA instead of RSA. ECC keys are efficient compared to RSA as RSA depends on multiplying two prime numbers to get a greater number is simple and factoring large numbers to return to the original primes is difficult. The usual ECC key size of 256-bits is equal to a 3072-bits RSA key, which is 10,000 times efficient than a 2048-bits RSA key to remain safe and to be ahead of a hacker's actions, RSA keys must be long and requires keys that are 2048-bits or longer, which

makes the process slower. The ECC uses simpler, smaller keys with consuming less energy to factor and convert more power to small mobile devices [17].

1.6 Mobile Computing

Mobile Computing is a technology that provides an environment that enables users to transmit data from one device to another device without the use of any physical link or cables. A computer or any wireless enabled device can transmit the data, voice and video. Also, it gives more flexible for the users to move from one location to another during the communication. Regardless of their place, it supports a wide variety of devices which allows people to access data and information. The mobile computing devices are mainly used in communication, education, directions, entertainment, business, healthcare and natural hazards with the support of internet to provide connectivity, social engagement and personalization. They are available in various sizes, i.e., notebooks, tablets, laptops, eReaders, handheld gaming devices, wearable devices and smartphones. The Wi-Fi (Wireless Fidelity) [18] is a wireless technology which can allow an electronic device to exchange data over the internet through radio waves. It is mainly available in institutions of higher learning, offices, restaurants, schools, recreational facilities, some public areas, and homes. Fig. 1.5 shows the connectivity of mobile devices.

Fig. 1.5: Mobile Connectivity

Programming languages are used for mobile system software. Operating System (OS) functions to run the software components onto the hardware [19]. Middleware components are used for deployment. Protocols and layers are used for transmission and reception. The programming languages used for mobile computing applications are *viz*., Java, J2SE, J2ME, JavaCard, J2EE, C, C++, Visual C++ and Visual Basic.

1.7 Operating Systems

Symbian OS, Window CE, Mac OS and Android OS are some of the OS [20] used in mobile computing applications. It offers the user to run an application without considering the hardware specifications and functionalities. They are used to schedule multiple tasks in a system. It provides user application‘s Graphical User Interface (GUI), Voice User Interface (VUI) components, and phone Application Programming Interface (API). It provides the device drivers for the keyboard, display, USB and other devices. The threats and issues of mobile computing can be divided into two categories such as general security issues and wireless security issues. The information and data residing on mobile devices have much more security issues and threats.

1.8 Mobile Communications

It refers to an infrastructure that ensures seamless and reliable communication among wireless devices [21]. The mobile communication consists of communication devices such as protocols, services, bandwidth and portals necessary to facilitate and support the stated services. These devices are responsible for delivering a smooth communication process. It can be classified into four categories such as fixed and wired, fixed and wireless, mobile and wired and mobile and wireless. Mobile hardware consists of mobile devices or device components that can be used to receive

or access the service of mobility (smart phones, laptops, portable PCs, tablet PCs, and Personal Digital Assistants (PDA)). These devices are inbuilt with a receptor medium that can send and receive signals.

Fig. 1.6: Mobile Device Hardware

These devices are capable of operating in full-duplex can send and receive signals at the same time. They don't have to wait until one device has finished communicating for the other device to initiate communications [22]. Fig. 1.6 shows the mobile device hardware.

Mobile software is a program that runs on mobile hardware. This is designed to deal capably with the characteristics and requirements of mobile applications. In other words, it is the heart of the mobile systems, and also an essential component that operates the mobile devices. Further, it provides portability which leads to wireless communication. Fig. 1.7 shows the mobile device software. It is noted that mobile devices are handheld devices. They have limited processing capability and less memory. If the existing exponentiation operation in performing encryption and decryption are used as it is, it consumes more time which will slow down the speed of mobile devices. To speed up the process, AC is used.

Fig. 1.7: Mobile Device Software

1.9 Addition Chain

An AC [23] for a positive integer n is a sequence, $1 = a_0 \leq a_1 \leq \cdots \leq a_r = n$ such that each member after a_0 is the sum of two earlier (not necessarily distinct) ones. It is noted that if the value of n is relatively small, the exact value of $l(n)$ is known. But, for large n, it is known that $l(n) = \log_2 n + \log_2 n\,(1 + \sigma(1))/(\log_2 n \log_2(n))$. An AC has elements, $1 = a_0 \leq a_1 \leq a_2 \leq \cdots \leq a_r = e$ with the property that for all $i > 0$ there exist $a_i = a_j + a_k$ and $r \geq i \geq j \geq k \geq 0$. An optimal AC [24][25] is the one which has the shortest possible length denoted by $l(e)$ and it is a strictly increasing sequence as duplicate chain elements could be removed to shorten the chain. For example, 1 - 2 - 3 - 6 - 12 - 13 is one of the optimal ACs for 13 and $l(13) = 5$.

In AC, there are two steps normally involved. They are addition and doubling steps [26][27] i.e., to get the next number (intermediate number) in AC, any two previous numbers are added together in addition step where as the current number is multiplied by two in doubling steps. To generate the AC for given n, two types of algorithms are normally used viz., deterministic and stochastic or bio-inspired. In deterministic algorithms since everything is deterministic and the optimal AC may not be obtained at all time. And also the length of AC is more for the integer n some times. Binary method, factor method, window method, sliding window method, Fibonacci method,

Lucas method, continued fraction method etc., are some examples of deterministic algorithm. But, bio-inspired algorithms (BIAs) [28] are based on animal or birds behavior and the optimal AC is not obtained by a single run and hence many more runs are needed.

1.10 Scope of Research

As the usage of mobile devices is increasing exponentially, the sensitive or potential information transmitted from one mobile to another should be protected. One of the most convenient modes of transmissions but it is associated with many security risks and the data can be intercepted while the transmissions are not being encrypted. Furthermore, all devices are not equipped with built-in security software and the users ignore the fact to install the security software. The common fact that the phones are vulnerable to threats or the exploitation by the cybercriminals. The security issues can be tackled by applying the cryptographic techniques using public-key algorithms to generate keys for both encryption and decryption to protect the information and maintain security from hackers. Due to the usage of keys, function codes and digital signatures' are widely used and is becoming more and more acknowledged as one of the best ways to secure data and applications both stores at rest and in motion between devices. More and more people are now using a mobile device in either personal or work related data. Today, the users were increasingly using unmanaged, personal devices for accessing sensitive enterprise information's and also establishing connections to the third party services beyond some security controls can leads to sensitive data to possible attackers.

Mobile devices have lightweight and very less in size. They have some challenges and certain limitations such as time, power battery power, memory size, processing speed,

screen size, resolution, etc. Providing security for mobile devices with limited power is a difficult job. ECC are normally used to transfer the date securely. They take more encryption and decryption time called operational time. If operational time is high, it consumes more battery power which ultimately it degrades the performance of mobile devices resulting in customer impatience and dissatisfaction. Thus, it is essential to speed of the process i.e., reducing the computational time of operational process, AC is used. There are many algorithms exist in the literature to generate the AC But, bio-inspired algorithms (BIAs) [29] have been proposed to generate the ACs. Thus, the objectives of the research is formulated as to generate the optimal ACs for the given integer n using

(i) Particle Swarm Optimization (PSO)
(ii) Simplified Swarm Optimization (SSO)
(iii) Bacterial Foraging Optimization (BFO)

The said algorithms are incorporated into RSA and ECC and they used in android as well as windows emulator and their performances are analyzed. They are explained in the next subsections in detail.

1.11 Particle Swarm Optimization

It [30] is a population-based optimization algorithm inspired by the motion of bird flocks and schooling fish and shares many similarities with evolutionary computation techniques. The system is initialized with a population of random solutions, and the search for the optimal solution is performed by updating generations. Unlike Genetic Algorithm (GA), PSO has no evolution operators, such as crossover and mutation. It has potential solutions, called particles, move in the problem space by following the current optimum particles and computationally more efficient in terms of both speed as well as memory requirements. It has become one of the most popular techniques

applied in various optimization problems, due to its ease and capability to find the optimal or near-optimal solutions.

1.12 Simplified Swarm Optimization

SSO [31], is a population-based algorithm to compensate for the deficiencies of PSO in solving discrete problems. This algorithm has recently been applied in many research areas because of its simplicity, efficiency, and flexibility. In SSO, each individual in the swarm, called a particle representing a solution, is encoded as a finite-length string with a fitness value. This scheme improves the Update Mechanism (UM), which is the core of any soft computing based methods in terms of convergence speed, energy and security. The UM updates each particle to be a compromise of those four sources, particularly a random movement, which is different from the original PSO, maintains population diversity, and enhances the capacity of escaping from a local optimum and achieved global optimum solution.

1.13 Bacterial Foraging Optimization

The concept of BFO was proposed by Kevin M. Passino [32][33]. It is a swarm intelligence based algorithm inspired by the behavior of forging based on the E. coli bacteria. It has four basic processes such as chemotaxis, swarming, reproduction and elimination-dispersal. In the elimination-dispersal process, a constant probability of elimination is assigned to all bacteria. The assignment is independent of bacteria's ranking in the population. Therefore, a bacterium which can be near to an optimal position may be replaced with one which is far away from the optimum solution, thus, affecting the convergence speed. The proposed algorithm uses a non-uniform probability distribution and it is implemented to replace the conventional methods.

The results shows the efficiency of the proposed algorithms in terms of convergence time, power and security.

1.14 Chapter Organization

The thesis is organized into eight chapters. The detailed study related to introductory concepts, types of cryptography, a brief introduction about RSA and ECC which are mainly used in mobile devices, and an introduction about the AC were discussed in chapter 1. It also discussed the introduction about the BIAs viz., PSO, SSO and BFO taken for AC generation. Further, this chapter also addressed the objectives and scope of the research clearly. Chapter 2 presents, various works related to mobile computing, various works related to swarm based algorithms viz., PSO, SSO and BFO. It also explains the works related to ACs.

The mathematical definitions of ACs, and need for EAs based ACs, brief introduction about RSA, various mathematical preliminaries required for ECC, ECC arithmetic and embedding the plaintext into EC points are discussed in chapter 3. It presents the need for security and various parameters taken in the work. It also highlights the comparative study of time taken for ET, DT, EP, DP and SE used in mobile cryptographic algorithm with and without incorporation of RM and RA in RSA and ECC respectively.

In chapter 4, the concepts of PSO, AC generation based on PSO are explained numerically. It also explains the usefulness of AC-PSO and AC-ECC. A comparative study between RSA, ECC and the proposed RM-RSA, RA-ECC, AC-PSO-RSA and AC-PSO-ECC are made and the experimental results obtained from them are analyzed.

The concepts used in SSO, reason for considering SSO for generating AC are present in chapter 5 with numerical examples. Once, the SSO based ACs are generated for the integers (encryption and decryption key for RSA, k[P] for ECC), they are incorporated into the mobile cryptographic algorithms RSA and ECC taken in this work. After implementing the said concepts, this chapter also analyzed the time taken for the said five parameters in this thesis.

As the BFO concepts play a vital role in generating ACs, chapter 6 presents the concept of BFO and the various processes used in it. The methodology used for generating the ACs using BFO is also explained numerically. After implementing the proposed methodology AC-BFO in RSA and ECC, the time taken for the said five parameters are analyzed too. It also presents the AC for some hard exponents and the comparative results obtained from the existing BIAs viz., GA, AIS, EP with the proposed BFO based AC for some integers.

The overall comparison of the existing RSA, ECC, RM-RSA, RA-ECC and the proposed AC-PSO-RSA, AC-PSO-ECC, AC- SSO-RSA, AC- SSO- ECC, AC- BFO-RSA and AC-BFO-ECC methods are present in chapter 7. Finally, the thesis ends with conclusion and future enhancements in chapter 8.

CHAPTER - II

REVIEW OF LITERATURE

2.1 Background

One of the basic principles used in mobile computing is broadcasting. It is radiated to everyone within the prescribed limit. The information transmitted within the limit in mobile devices must be protected. This is possible only by providing the security for it so that ithe information passed through the communication channel could be protected from various attack by the attacker. Thus, it is essential to take utmost care to prevent the attack while the user roams through different networks with heterogeneous security infrastructure. One way is to change the information in different form i.e., non-meaningful form. In order to achieve is to use cryptographic algorithms which are mainly used in mobile devices. Among them, RSA and ECC public-key cryptographic algorithms play a vital role in performing security. The basic building blocks of ECC is scalar point multiplication k[P] where k is a scalar and P is a point on EC. Similarly, in RSA the encryption and decryption is of the form x^e mod n where e is encryption / decryption key and x is plaintext/ciphertext. Normally exponentiation operation takes more time than multiplication which takes more time than addition and subtraction.

To reduce the time, exponentiations and multiplications are performed by RMs and RAs respectively. To reduce it further AC is used. In order to generate the AC, the concept BIAs like PSO, SSO and BFO are applied and their results are discussed. Using them, the ACs for the exponent of RSA and k[P] of ECC are generated. The encryption/decryption time and energy required for encryption/decryption [37] are

also computed and the performances of the cryptographic algorithms in mobile devices are analyzed with and without the incorporation of AC.

2.2 Review of Works Related to Mobile Computing

Mavridis I. and Pangalos G.[34], discussed the operational, security issues of mobile components in distributed environments and also gave the details about the elimination of intrinsic problem in wireless networking using mobile agents. They have implemented a healthcare paradigm, with security mechanisms. Erik Olson and Woojin Yu [35] surveyed various symmetric-key algorithms viz., RC5, RC6, Twofish and Triple-DES and their usage in mobile computing, specifically in the Palm Pilot, which use the Motorola's Dragon Ball-EZ processor. They illustrated that the architecture used in the processor was similar to the 68K processor and it did not provide the power and versatility of current processors.

Wendy Chou [36], surveyed the explosive growth in the usage of mobile and wireless devices demands a new generation of Public-key Cryptography (PKC) schemes, and the limitations on power, bandwidth to provide security in mobile devices, use of ECC, its security, performance and also its applications. Limor Elbaz [37], implemented PKC in security of wireless devices and the use of Public-key Infrastructure (PKI) in applications of mobile phones and proved the Discretix Crypto Cell implementation of cryptographic algorithms to enable wireless devices to become PKI enabled cum efficient, lightweight and standard-compliant. Dharma P. Agrawal et al. [38], discussed the technology in mobile computing users by combining wireless networking and mobility which served anytime and anywhere with of various new applications and also services. They also analyzed some security issues and various threats and concluded that encryption played an important role for secured communication in mobile computing environments.

Hanping Lufei and Weisong Shi [39], discussed the emergence of heterogeneous devices, diverse networks and the difficulty in using a one-size-fits-all encryption algorithm. They proposed an adaptive encryption protocol to choose a proper encryption algorithm dynamically to enhance the security from the candidate algorithms and minimized the time overhead. Abhishek Kumar Gupta [40], discussed the need for information as a driving force for the incoming growth in web technology, wireless communication and portable computing devices and also explained the field of mobile computing (computing and communication) with aim of providing seamless computing environment for mobile users.

S. Krishna Mohan Rao and Dr. A Venugopal Reddy [41] discussed the data dissemination to access the data item quickly in mobile devices with minimum access time so that the mobile clients saved the precious battery power while using resource-limited Wireless Sensor Networks (WSN), with reliable and efficient security mechanisms. Using two potential block ciphers, RC5 and AES-Rijindal discussed and analyzed the suitability of the algorithm for resource-limited wireless network security by M. Razvi Doomun and KMS Soyjaudah [42].

Kar and Banshidhar Majhi [43] proposed an efficient password security of Multi-Party key exchange protocol based on Elliptic Curve Discrete Logarithm Problem (ECDLP) and the protocols allowed a group of parties communicating over a public network to establish a common secret-key called session-key and also built a protocol for password authentication model, where group members were assumed to hold an individual password rather than a common password with two one-way hash functions to build the security level high. Mooseeop Kim et al. [44], proposed a compact architecture for a cryptographic engine on a mobile platform which had very stringent limitations with respect to the circuit area and the consuming power. It was

highly effective to implement the scalable RSA and unified SHA algorithm with a minimum resource usage.

Bruno P.S. Rocha et al. [45] demonstrated a security service as a middleware to dynamically change the security protocols used between two peers with variations on wireless medium parameters, usage of system resources, hardware resources, application-defined Quality of Service (QoS) metrics and desired data security levels. Sathish Alampalayam Kumar [46] suggested a mobile agent based mobile computing system, various types of security attacks and security solutions. Sameer Hasan et al.[47] proposed a non-server (that is P2P) PKC architecture to secure the mobile communications and implemented various security services needed for mobile communication. They used NTRU algorithm for public-key cryptography in non-server architecture and tested on real equipment, the solution security and potential risks.

Rahat Afreen and S.C. Mehrotra [48] discussed the ECC and its proper implementation to analyze in hardware as well as software platforms. Helena Rifa-Pous and Jordi Herrera- Joancomarti [49] discussed the performance of different cryptographic algorithms in PDAs and compared it with device's costs in terms of OS, screen, network interfaces to determine the overhead and the results were used to estimate the costs of network security protocols design. Jagdish Bhatta and Lok Prakash Pandey [50] proposed a software level cryptographic protocol implementation to measure the energy level through the device's serial port. They found the proposed protocol proved better security and less consumption of energy than the existing cryptographic protocols.

K. Sathish Kumar et al. [51] explained the mobile hand-held device in an efficient way to deliver real time data to users and implemented an energy efficient authentication protocol with a high level security with minimum energy consumption for mobile devices. In 2012, Masoud Nosrati et al. [52] proposed an algorithm made the data into unreadable text which could be decoded only with the associated key and consumed less CPU time, memory, battery power and computation time in various mobile devices with their OSs. Ravinder Singh Mann et al. [53] presented the comparative analysis of ECC, AES and RSA algorithms experimentally with parameters such as computation time and complexity of the algorithms and concluded that ECC has more complexity in mobile devices.

Giripunje et al. [54] provided effective security solution using PKC implementation in two parts: first part was designing API for ECC to generate the shared key for secure communication and the second part dealt with a creation of web service which distributed this key to validate the mobile user. Ameya Nayak [55] discussed the growing android community, its malware attacks, security concerns, aid in serving as the continuous challenges of identifying current, future vulnerabilities as well as incorporating security strategies against them and this focus on mobile devices. Srikanth Pullela [56] has proposed protocols for various applications like wireless application protocol and mostly based on the public and private-key cryptography. V. Gayoaso Martinez and L. Hernandez Encinas [57] have discussed the ECC was one of the best options for protecting sensitive information. The latest version of the JAVA platform was a cryptographic provider - SunEC which was implemented for EC operations and protocols. They explained the applications for generation of key pairs, perform key exchanges and produce digital signatures with EC in JAVA.

Muhammad Waseem Khan [58] explained the Short Message Service (SMS) which was used in mobile services of GSM networks but this facility was not achieved secure transmission of plaintext between different mobile phone devices. However, SMS did not have its own built-in mechanism to secure the transmitted data because security was not considered as a priority application for mobile devices. The existing schemes provided a room for the secure SMS message communication. The effect of each security scheme on mobile device's performance was also observed.

Ram Ratan Ahirwal and Manoj Ahke [59] explained the Diffie-Hellman scheme as one of the key exchanging cryptosystem and no messages were involved in this scheme. Two different methods to encrypt and decrypt the message were proposed by them. They pointed out that the second method support the system with more security than the first method because the sender computed the exponentiation function between the coordinates of the encryption algorithm and the receiver computed the inverse of the exponentiation function between the coordinates of the key in the decryption algorithm, While in the first method, the sender computed the multiplication between the coordinates of the key in the encryption algorithm and the receiver computed the multiplication between the coordinates of the key in decryption algorithm and forward secrecy in HTTPS protocol.

Sathish Kumar et al. [60] have discussed the mobile hand-held devices which were used in an efficient way to deliver real time data to the users in the battle field military applications and the use of security features such as data confidentiality, authentication etc., which were not readily offered by mobile environment. They have proposed the implementation of energy efficient authentication protocol for mobile devices. Hamed Khiabani et al. [61] explained the extensive deployment of

wireless networking, mobile, embedded devices and other pervasive computing technologies that were prone to security threats for which nobody would be prepared for them. Security and privacy were the main concerns in mobile computing which could be observed from several perspectives including hardware, operating systems, networks, databases, user interfaces, and applications.

Seema P. Nakhate and R.M. Goudar [62] have implemented a secured password based mutual authentication protocol for client-server computing using ECC framework which provided secure communication between client and server with the help of user email-id and mobile phone authentication device for mobile handheld device since it could be best suited for constrained resources such as computational power, storage capacity. They were extremely limited especially devices like Mobile phones, PDA's, Palmtops and Smart cards. Vishnu V and Shobha R [63] discussed the security in Wireless Sensor Networks (WSN). They have applied dynamic election of Cluster Head (CH) mechanism and two evolutionary approaches SET-IBS and SET-IBOOS, since it provided security in data transmission and reduced data losses due to nodes failure, less residual energy selected in CH. It improved the lifetime of network by increasing the time of FND (First Node to die).

Tanmoy Kumar Bishoi et al. [64] proposed an algorithm to encrypt the data using symmetric-key encryption with variable length key size. In [65], Sujithra M et al. explained the high performance computing techniques. They also implemented cryptographic algorithms and tested in Local as well as Cloud environment. They have revealed that storing mobile data in cloud increasing efficiently and AES algorithm performed better when compared with other algorithms with respect to mean processing time but the combination of MD5+ECC+AES algorithms qualify

better than Speed-Up ratio. Said Bouchkaren and Saiida Lazaar [66] discussed secure data transmission through Internet. They have designed and implemented a new secret-key cryptosystem due to a number of iterations of encryption and decryption of data in blocks, using cellular automata and compared them with AES algorithm. Also proved that the new algorithm resisted against statistical attacks, faster than AES-256, achieved good confusion and diffusion tests.

2.3 Review of Works Related to PSO

Arbit and Ashwini Kumar [67], suggested Optimized ECC (O-ECC) to assist more secure and improved protocol design with easy computation mathematically. In [68], Ahmed Tariq Sadiq discussed the PSO. The benefit of mutation in PSO (MPSO) was used as momentum and diversity tool in the population. Experimental results clearly showed that the amount of recovered key of classical ciphers and fitness function values were better than PSO. Ahmed A. Esmin and Germano Lambert-Torres [69] have proposed a methodology which was used to determine the control variable settings for real power loss minimization in the transmission system. It employed the PSO algorithm for the optimal setting of Optimal Power Flow (OPF) based on Loss Minimization (LM) function, tested on IEEE 14, 30, 118 Bus systems and the results were compared.

G.Prakash and Dr.M.Kannan [70] discussed that the cryptographic smart cards were used for most of the online transactions. They have designed an integrated approach of cryptography and steganography which could be used for smart card security. Initially, user’s confidential details were encrypted using the most secure ECC technique and then the encrypted cipher was embedded into the users 'photographic image using steganography named Optimized Modified Matrix Encoding (OMME) algorithm. Cuevas et al. [71] proposed the swarm intelligence models with collective

behaviour in swarms of insects or animals called the social spider optimization for solving optimization tasks. The outcome revealed a high performance for searching a global optimum with several benchmark functions. Wilayat Khan et al. [72], discussed the mobility which was one of the major features of wireless communication systems and handheld devices form a major part of the systems. The limited resources like battery, memory and computational power of these devices was a bottle neck in the security of such devices was also discussed.

Rangit j. Bhosale et al. [73] proposed the Mobile Ad-hoc Networks (MANET) in wireless technology, having features like dynamic topology and self-configuring ability of nodes. Swapna B. Sasi and N. Sivanandam [74] compared and analyzed the performance level with various parameters such as number of keys stored, battery capacity, runtime. They also concluded that high storage and energy were required for storing the keys. Dolly U. Jeswani et al. [75], discussed the cryptographic algorithms which were the key factor of the security mechanisms used for data storage and uninterrupted network transmissions. A PSO oriented cryptanalysis technique for breaking the key used in AES algorithm was also introduced. Swarm Intelligence based cryptanalysis provided a best and optimized solution. Chia-Ling Huang and Wei-Chang Yeh [76] demonstrated to optimize the Reliability Redundancy Allocation Problems (RRAP) for the series-parallel system, the complex (bridge) system and the over speed protection of gas turbine system. To solve the RRAP, PSO algorithm was proposed to improve the computation efficiency and found that it outperformed the previously best-known solutions.

Ji Weidong and Zhu Songyu [77] discussed as PSO was the most common algorithms for optimization because of its simple, convenient and good robustness. They have proposed a new particle swarm algorithm as improved cut PSO algorithm based on

filtering mechanism (ELPSO) to improve its operational speed and more accurate. Jin Yang et al. [78] presented the enhanced version of the network performance of WSNs with mobile sinks (MWSNs) in an efficient routing strategy using PSO to build the optimal routing paths. A novel greedy discrete particle swarm optimization with memory (GMDPSO) was introduced to improve the greedy forwarding routing, a greedy search strategy was designed to drive particles to find a better position quickly, searching history was memorized to accelerate convergence. Simulation results revealed that the new protocol significantly improved the robustness and adapted to rapid topological changes with multiple mobile sinks, while efficiently reducing the communication overhead and the energy consumption.

2.4 Review of Works Related to SSO

Joppe W et al. [79] have explored the deployment of ECC in practice by investigating its usage in Bitcoin, SSH, TLS and the Austrian citizen card. They concluded that the researchers and developers could identify the threats discovered from the attackers and track the implementation problems to improve the security of the cryptographic protocols and libraries. The commonly used Itoh-Tsujii algorithm (ITA) was used to compute the inversion by an entirely sequential process consisting of multiplications and squarings. Lijuan Li and Shuguo Li [80] proposed a modified ITA algorithm (MITA) for inversion with polynomial basis (PB) which could reduce the required clock cycles of ITA by enabling the parallel computation between part of multiplications and squarings. Furthermore, they were generalized using ACs and to find the optimal addition chains (OACs) leading to the fastest inverters with given hardware resources.

An EC addition law is said to be complete if it correctly computes the sum of any two points in the EC group. One of the main reasons for the increased popularity of Edwards curves in the ECC community was that they could allow a complete group law that was also relatively efficient (e.g., when compared to all known addition laws on Edwards curves). Such complete addition formulas could simplify the task of an ECC implementer and at the same time, it greatly reduced the potential vulnerabilities of a cryptosystem. Unfortunately, until now, complete addition laws that were relatively efficient have only been proposed on curves of composited order and have thus been incompatible with all of the currently standardized prime order curves.

Joost Renes, Craig Costello and Lejla Batina [81] have presented optimized addition formulas to complete on *every* prime order short Weierstrass curve defined over a field *k* with char(k)≠2,3char(k)≠2,3. Compared to their incomplete counterparts, these formulas required a larger number of field additions, but interestingly it required fewer field multiplications. Furthermore, they have discussed how these formulas could be used to achieve secure, exception-free implementations on *all* of the prime order curves in the NIST (and many other) standards. PSO could locate the region of the optimum faster than EAs, but once in this region it progressed slowly due to the fixed velocity stepsize [82]. Almost all variants of PSO tried to solve the stagnation problem.

Bioinspired algorithms have been employed in situations where conventional optimization techniques could not find a satisfactory solution. For example, when the function to be optimized was discontinuous, nondifferentiable, and/or present too many nonlinearly related parameters [83]. One of the most well-known bioinspired algorithms used in optimization problems is PSO, which basically

consisted of a machine-learning technique loosely inspired by birds flocking in search of food. More specifically, it had a number of particles that collectively move on the search space in search of the global optimum [84].

Wei-Chang Yeh et al. [85] formulated a General Multi-level Redundancy Allocation Problem (GMRAP) to break the restrictions and generalize the above problems. Furthermore, a novel algorithm called SSO with Modular Search (SSO-MS) was proposed to solve the GMRAP. Finally, the results obtained by SSO-MS were compared with those obtained from GA and PSO. The comparative results showed that the proposed SSO-MS was most promising among three algorithms and demonstrated the effectiveness. Asymmetric cryptographic algorithms were a robust technology used to reduce security threats in the transmission of messages on the network. Nowadays, one of the disadvantage is the mathematical solutions because they require a greater amount of calculation that led to the need for increased use of computational resources. Fausto Meneses et al. [86] developed an algorithm to optimize the RSA encryption algorithm and to improve the security, integrity and availability of information. The results showed that the efficiency and functionality of the RSA algorithm in terms of information security. Also, the parameter such as time, memory, processor and network performance were analyzed while performing encryption and decryption were lower than other RSA solutions, because calculations was performed on the client and server.

Nigel P. Smart [87] overviewed about ECs with modern public-key systems. It provided improved efficiency and bandwidth. Katz and Mazur [88], presented with mathematical proof of recent developments in ECs with their moduli spaces and they began with Jacobi's "Fundamenta Nova" in 1829, and the modern theory was erected by Eichler-Shimura, Igusa, and Deligne-Rapoport.

In [89], the authors explained the power of algebra as generalised arithmetic and it led to a task in which others could explore other possible relations made with a similar process. They insisted that those who designed and would inspire further investigation and generalisation. The problem of finding the shortest AC for a given exponent is of great relevance in cryptography, but it was also very difficult to solve since it is an NP-hard problem. Stjepan Picek et al. [90] proposed a GA with solutions with new crossover and mutation operators to minimize the length of the ACs corresponding to a given exponent. The results were compared with respect to those generated by other meta heuristics for instances of moderate size, values up to $2^{127}-3$. Furthermore, three additional strategies were adopted and the results indicated that the proposed approach was very promising alternative to deal with this problem.

Finding the shortest AC for a given exponent is a significant problem in cryptography. Crossover and mutation operators of GA to minimize the length of the ACs corresponding to a given exponent. Stjepan Picek et al. [91] developed a repair strategy that gives significant enhanced performance and the results(values up to 2^{255}-21) were compared with respect to those generated by other metaheuristics for exponents of moderate size and optimize the ACs with regards to the type of operations as well as the number of instructions required for the implementation. AC calculations play a critical role in determining the efficiency of cryptosystems based on isogenies on elliptic curves. However, finding a minimal length AC is not easy. A generalized version of the problem, in which one must find a chain that simultaneously forms each of a sequence of values, is NP-complete. For the special primes used in such cryptosystems, finding fast ACs for finite field arithmetic such as inversion and square root is also not easy. Brian Koziel et al. [92] investigated that the shape of smooth isogeny primes and proposed new methods to calculate fast ACs.

Furthermore, they have provided techniques to reduce the temporary register consumption of large exponentials, applicable to both software and hardware implementations utilizing ACs. Finally, the procedures were compared multiple isogeny primes by the complexity of the ACs.

A novel graph based methods have been proposed [93] for generating the optimal AC where the vertices of the graph representing the numbers used in the AC and edges representing the movement from one number to another number in the AC. They proposed two methods, Method 1 termed as GBAPAC which generated all possible optimum ACs for the given integer n by considering the edge weight of all possible numbers generated from every number in AC. Method 2 termed as GBMAC, not all possible numbers were generated from the particular number in forming AC since they are mutually exclusive. That is, only one number is generated by doubling step and the rest of the numbers are generated using addition step.

A methodology [94] was presented for converting the inner dynamics of PSO algorithm into complex network for improving the performance of evolutionary computational techniques. It could be used for adaptive measures to manage and also to put significant amount of information about the inner dynamics of PSO algorithm into a complex network. A new BIA, namely Bird Swarm Algorithm (BSA), was proposed [95] for solving optimisation problems. BSA was based on the swarm intelligence extracted from the social behaviours and social interactions in bird swarms. Birds mainly have three kinds of behaviours: foraging behaviour, vigilance behaviour and flight behaviour. Birds might forage for food and escape from the predators by the social interactions to obtain a high chance of survival. By modelling these social behaviours, social interactions and the related swarm intelligence, four search strategies associated with five simplified rules were formulated in BSA.

Simulations and comparisons based on eighteen benchmark problems demonstrated the effectiveness, superiority and stability of BSA.

In order to improve the particle swarm optimizer (PSO) for solving complex multimodal problems, an improved PSO with full information and mutation operator (PSOFIM) was proposed in [96]. In PSOFIM, a novel mutation was adopted to improve the history optimal position of particle (pbest) by disturbance in operation of each dimension. Additionally, a full information strategy for each particle was introduced to make the best use of each dimension of each particle to ensure the information utility for swarm topology where each particle learnt from its neighbourhood information for it optimal position to improve itself study ability, whose strategies improve the swarm fly to the probability of the optimal solution. The simulation experiment results of benchmark function tests showed that PSOFIM has better performance than the basic PSO algorithm.

In supply chain management, reducing operating cost and satisfying customer demand are the most important things. However, the products may be spoilt during the delivery due to collisions, traffic accident, weather factor, theft and so on. Hence, the authors [97] considered the deterioration effect in a three-stage supply chain deteriorated network with a mathematical model. A novel AI algorithm named SSO was adapted in the above problem to minimize the total operating cost. Extending local search (ELS) was attached to enhance the performance of the original SSO. A numerical example of network system was presented to compare the proposed algorithm with GA and PSO. Results indicated that SSO-ELS provided a better solution than its competitors.

Predictive analytics analyze the present and the historical information's and make future predictions utilizing data mining or machine learning techniques. Predictive models usually check for some patterns and relationships leading to certain behaviours based on the dependent variables. In [98], the author proposed a mechanism named analysis and prediction of application usage (APAU) in android phones for providing recommendations to a smart phone user while selecting applications of their interest like mail checking, messaging and making calls. APAU mainly focused on identifying usage patterns and investigating the human behaviour during application selections by extracting the generic behavioural patterns to predict and provide useful set of recommendations. Simulated real-world interaction with a device and test the features by using the tools included with microsoft emulator [99] for windows 10 mobile. The emulator like a desktop application that emulated a mobile device running windows 10. It provided a virtualized environment in which it could be used to debug and test windows apps without a physical device. It also provided an isolated environment for application prototypes. The emulator was designed to provide comparable performance to an actual device. This could be used to simulate real-world interaction with a device and test various features by using the tools included in the Microsoft Emulator. In [100], the elementary theory with the concepts of ECs, facts, cryptosystem and factorization were discussed.

In [101], analog based on ECs over finite fields of public-key cryptosystem was discussed, which used the multiplicative group of a finite field. These EC cryptosystems might be more secure, because the analog of the DLP on ECs are likely to be harder than the classical DLP especially over GF(2"). Furthermore, the question of primitive points on an EC modulo p, and gave a theorem on non smoothness of the order of the cyclic subgroup generated by a global point. In [102], the authors

discussed about the use of ECs in cryptography and proposed an analogue of the Diffie-Hellmann key exchange protocol which appeared to be immune from attacks of the style of Western, Miller, and Adleman. With the current bounds for infeasible attack, it was 20% faster than the Diffie-Hellmann scheme over GF(p). As computational power increased, this disparity should get rapidly bigger.

2.5 Review of Works Related to BFO

ECC [103] [104] pairings in pairing-based cryptosystems and computing isogenies in the quantum-resistant isogeny-based cryptosystems was discussed. To get the next number, there are two steps normally used in AC. They are addition and doubling steps, i.e., to get the next number (intermediate number) in AC, any two previous numbers are added together in addition step, whereas in the doubling step, the current number is multiplied by two. To generate the AC for given n, two types of algorithms are normally used viz., deterministic and stochastic or bio-inspired evolutionary algorithms. Kevin M. Passino proposed it in 2000, and it has been widely accepted as a new nature-inspired optimization algorithm. It is inspired by the social foraging behavior of Escherichia Coli, i.e., a bacteria present in the human intestine and has drawn many researcher's attention.

In [105], Hugo Volger presented several results on l(n). In particular, they determined l(n) for all n satisfying l(n) ≤ 3 and proved $\lfloor log\ n \rfloor + 2 \leq l(n)$ for all n satisfying $s(n) \geq 3$, where $s(n)$ is the extended sum of digits of n. In [106], Y.H. Tsai and Y.H. Chin found some mathematical properties of the shortest-length AC for certain integers whose binary patterns met some special forms; and the correctness of these properties was proved. In [107], Bergeron et al. proposed generating the shortest AC based on the continued fraction. They gave a general upper bound for the

complexity of continued fraction methods as a chosen strategy function. Thus, the total number of operations required for the generation of an AC for all integers up to n was shown to be ($n\ log^2 n\gamma n$), where $n\gamma n$ is the complexity of computing the set of choices corresponding to the strategy and proved an analogy of the Scholz-Brauer conjecture.

F Bergeron et al. [108] generated a method of fast ACs for positive integer n, using continued fraction up to 1000 number obtained with optimal length (with 29 exceptions optimal length plus one). A computer could generate a random sequence of numbers [109], $U_0, U_1, U_{2,}$ ...that behaves as if each number was independently selected at random between 0 and 1 with the uniform distribution. A new algorithm of optimal ACs was generated in [110] and also faster than the best-known methods. It was applicable for single values and slower than the best-known methods. It did not require any pre-computed values and it was considered suitable for finding optimal ACs for point values.

Bounds on sums of ACs and properties of optimal ACs were discussed in [111]. The study exhibited that the final step in an optimal AC of an even number always have doubling, and also the sum of an optimal AC for an odd number n is asymptotically nearly $5n^2$. Noboru Kunihiro and Hirosuke Yamamoto [112] developed two systematic methods viz., run-length encoding (RLE) and hybrid for generating short AC. They proved that the hybrid method was far better than RLE with a reduced 8% of the AC length.

Nareli Cruz - Cortéset et al. [113] explored the usage of a GA approach for the problem of finding optimal (shortest) ACs for optimal field exponentiation computations. The GA heuristic presented in this work was capable of finding almost

all the optimal ACs for any given fixed exponent e with $e < 4096$. They found that GA strategy's percentage error was within 0.4% of the optimal for all cases considered. In other words, for any given fixed exponent $e$with $e< 4096$, they found that strategy was able to find the requested shortest AC in at least 99.6% of the cases. N. Cruz- Cortés et al. [114] proposed an AIS to generate an optimal AC. In that paper, they dealt with the optimal computation of finite field exponentiation, which was a well-studied problem with many important applications in error-correcting codes and cryptography.

Raveen R. Gounder et al. [115] discussed a new strategy for doubling-free (SPA-resistant) short addition-subtraction chain(GRASC) for an arbitrary integer by using a precise golden ratio. In this, 12% to 28% reduction was obtained in the average chain length compared to other doubling-free AC methods. Alejandro Le´on - Javier et al. [116] discussed the PSO algorithm to find the shortest ACs with different exponents. Mohamed M. Abd. Eldayamet al. [117] proposed an algorithm for shorter AC based on the window method with small width using 2's complement. They proved that the proposed algorithm was more efficient than the last result with a 20% minimum. S Domínguez-Isidro and E Mezura-Montes et al. [118] proposed an algorithm using EP to find the minimal length AC and the results obtained were more promising than the other nature-inspired meta heuristic approaches but with a lower number of evaluations per run. The proposed EP algorithm comprised the solution encoding with suitable fitness function and initial population, a mutation operator, and the survivor selection mechanism and EP did not use other operators such as crossover nor additional mechanisms like parent selection in GAs. In [119], a note an AC was presented. Niel Michael Clift [120] proved the perfect matches in the Scholz–Brauer conjecture $l(2n-1) = l(n) + n - 1$ for new values. The minimal sequence of minimal

multiplications required for performing modular exponentiation using Brauer Chains' concept by GA was discussed in [121].

K. Mani [122] proposed division based AC to generate the optimal ACs for the small exponents, exactly matched with ACs generated by the latest methods. But, for some large exponents, there was a very small increase in chain length (at most three).

P. Anuradha Kameswari and B. Ravitheja [123] derived a Lucas AC for any integer n to obtain Lucas sequence $(a, 1)$ and also proved that the computation of $V_n(a, 1)$ using this Lucas AC is based on $V_{x+y}(a, 1)$ for $x, y, x–y$in the Lucas AC. Stjepan Piceket et al. [124] derived that the GA approach with an novel encoding using crossover and mutation operators to minimize the length of the ACs with respect to a given exponent. Aaron Hutchinson and Koray Karabina implemented algorithms [125], for multidimensional differential ACs and applied these chains to ECC. This algorithm has the unique key features using n dimension. With key efficiency cum security features like uniformity, parallelized, and differential addition formulas were adopted by allowing speed using precomputation cost and storage requirements.

Dustin Moody and Amadou Tall [126] derived minimal chains with low Hamming weight using addition-subtraction chains with Lucas addition-subtraction using $\ell(n)$ the minimal length n, and proved that $|\ell-(2n) - \ell-(n)| \leq 1$ for all integers n of Hamming weight ≤ 4 to have arrived a conclusion that minimal addition- subtraction chains for low Hamming weight integers, with the consideration of odd integers. In [127], the authors implemented a new parallel algorithm to obtain minimal AC for n. The experimental studies on multicore systems revealed that this algorithm's run time worked faster than the sequential one and obtained the maximum speed up of 2.5 times than the best known sequential algorithm.

Narendra Mohan [128] discussed in wireless sensor networks (WSNs) to enhance the network lifetime and minimize the energy consumption in sink nodes contained additional resources like long-range antenna, powerful batteries, large memory. This should be achieved using Enhanced Emperor Penguin Optimization (EEPO) algorithm. Bacteria foraging is one of the optimization and evolutionary algorithms. Kevin M. Passino proposed it in 2000, and it has been widely accepted as a new nature- inspired optimization algorithm.

2.6 Chapter Summary

From the above literature, it is understood that there is a need to develop enhanced supportive method to protect from hackers. Moreover, the devices have limited battery power, storage and have some security threats while transferring sensitive information's through mobility as well as wireless devices like mobile devices. Hence, the five parameters have been taken in this research viz., encryption, decryption time, encryption, decryption power and security. This is possible only with AC. Even though, too many methods exist in literature for generating AC it is found from the literature that ACs were not generated by BIAs viz., PSO, SSO, BFO with incorporation into RSA and ECC. These motivate me to select these methods to generate the ACs and they are discussed in the subsequent chapters.

CHAPTER – III

OVERVIEW OF ADDITION CHAIN AND MOBILE CRYPTOGRAPHY

3.1 Background

Mobile computing works on the principle of broadcasting so that the information is radiated to everyone within the wave range to increase the security threats and cyber attacks replicated quickly and easily. Thus, care must be taken in handling for those types of attacks to provide information security while the user roams through different networks with heterogeneous security infrastructure. For that several cryptographic techniques are employed in mobile devices. Among them, the public-key cryptographic algorithms like RSA and ECC play a vital role in performing security. The basic building blocks of ECC are scalar point multiplication k[P] where k is a scalar and P is a point on EC. Similarly, in RSA, the encryption and decryption is of the form $x^e mod\ n$ where e is encryption/decryption key. Normally, exponentiation operation takes more time than multiplication which takes more time than addition and subtraction. To reduce the time, exponentiations and multiplications are performed by repeated multiplications (RMs) and repeated additions (RAs) respectively. To reduce the time further AC is used.

3.2 Mathematical Definitions of Addition Chain

This section describes some important definition of AC.

3.2.1 Definition (Addition Chain)

An AC for a positive integer n is a sequence, $1 = a_0 \leq a_1 \leq \cdots \leq a_r = n$ such that each member after a_0 is the sum of two earlier (not necessarily distinct) ones. The number $l(n)$ is called the length of the AC. It is noted that if the value of n is

relatively small, the exact value of $l(n)$ is known. But, for large n, it is known that

$$l(n) = \log_2 n + \log_2 n\,(1 + \sigma(1))/\log_2(\log_2 n) \quad \text{... (3.1)}$$

An AC also defined as a finite sequence of positive integers called elements,

$$l = a_0 \leq a_1 \leq a_2 \leq \cdots \leq a_r = e \quad \text{... (3.2)}$$

with the property that for all i $>$ 0 there exist a_j , k with

$$a_i = a_j + a_k \; and \; r \geq i \geq j \geq k \geq 0 \quad \text{... (3.3)}$$

3.2.2 Definition (Optimal AC)

An optimal AC is the one which has the shortest possible length denoted by *l(n)* and it is a strictly increasing sequence as duplicate chain elements could be removed to shorten the chain. For example, 1–2–3–6–12-13 is one of the optimal chains for 13, and its l(13) = 5.

The construction of each element of an AC [24] is called a step.

For an AC, $l = a_0 \leq a_1 \leq \cdots \leq a_r = n$, the following steps are involved.

Doubling step: $a_i = 2a_{i-1},\ i > 0.$

Non-doubling step: $a_i = a_j + a_k,\ i > j > k \geq 0.$

The steps of the form $a_i = 2a_j, j \leq i - 2$ are defined as non-doubling steps.

Big step: $\lambda(\ a_i) = \lambda(a_{i-1}) + 1.$

Small step: $\lambda(\ a_i) = \lambda(a_{i-1}).$

Thus, length of the AC, $l(n)$ can be split into two components as $l(n) = \lambda(n) + \delta(n)$

From the above, it is understood that the first step is always a doubling step. A doubling step is always a star step and never a small step. A doubling step must be followed by a star step. If step i is not a small step, then step $i + l$ is either a small step or a star step, or both. It is noted that, not all doubling steps are big steps but big

steps are always doubling. Because l (n) is fixed for a given positive integer, finding optimal ACs amounts to minimizing the number of small steps across all possible chains. It is noted that for the given integer n, more number of ACs are possible. But, for finding at least one of the shortest AC is an NP-hard problem. For example, n=170, all possible optimum ACs are listed in fig. 3.1.

1-2-3-5-10-20-40-45-85-170	1-2-3-5-10-20-40-80-85-170	1-2-3-5-10-20-40-80-90-170
1-2-3-5-10-20-40-80-160-170	1-2-4-5-10-20-40-45-85-170	1-2-4-5-10-20-40-80-85-170
1-2-4-5-10-20-40-80-90-170	1-2-4-5-10-20-40-80-160-170	1-2-4-6-10-20-40-80-90-170
1-2-4-6-10-20-40-80-160-170	1-2-4-8-9-17-34-51-85-170	1-2-4-8-9-17-34-68-85-170
1-2-4-8-9-17-34-68-102-170	1-2-4-8-9-17-34-68-136-170	1-2-4-8-10-20-40-80-90-170
1-2-4-8-10-20-40-80-160-170	1-2-4-8-16-17-34-51-85-170	1-2-4-8-16-17-34-68-85-170
1-2-4-8-16-17-34-68-102-170	1-2-4-8-16-17-34-68-136-170	1-2-4-8-16-18-34-68-102-170
1-2-4-8-16-18-34-68-136-170	1-2-4-8-16-32-34-68-102-170	1-2-4-8-16-32-34-68-136-170

Fig. 3.1: Optimum Addition Chains for n = 170

3.2.3 Definition (Brauer Chain)

A Brauer chain is an AC that always uses the previous value for the next one. In other words, it is a sequence of integers $a_0, a_1, \dots, a_r$ with $a_0 = 1$, $a_r = n$, such that $a_i = a_j + a_{i-1}$, i.e., a Brauer chain is an AC in which every member after the first is the sum of the immediately preceding element and a previous element (possibly the same element). For example, $1 - 2 - 3 - 6 - 7 - 13$ is a Brauer chain for 13.

3.3 Need for Evolutionary Algorithms Based Addition Chains

To generate the ACs for an integer n, two algorithms are broadly used. They are deterministic and evolutionary algorithms (EAs). In deterministic algorithms, since everything is deterministic and the optimal AC may not be obtained at all time. Binary method, factor method, window method, sliding window method, Fibonacci method, Lucas method, continuous fraction method etc., are some examples of

deterministic algorithm. EAs or Bio-Inspired Algorithm (BIAs) are inspired by the idea of either natural evolution or social behavior of insects or birds [29][30]. The optimal ACs produced by EAs are not obtained by a single run. Many more runs are needed to obtain optimal AC which will eventually take more run. Some examples of EAs are Genetic Algorithm (GA), Artificial Immune System (AIS), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Simplified Swarm optimization (SSO). Bacteria Foraging Optimization (BFO) etc. In this thesis, three popular BIAs viz., PSO, SSO and BFO algorithm concepts are taken and they are discussed in next chapters.

3.4 Reason for Taking RSA and ECC

The usage of mobile devices are growing rapidly is the urge of today's scenario. The applications of smart mobile phones are also increasing which lead to many security issues too. The security features can be taken and applied on these devices. Mobile devices deal with heterogeneity of networks and also in ubiquitous intelligent environment with embedded computers everywhere and reliable services to the user in an easy way. Even though, they have more offerings to the user, lot of challenges like disconnection, low/ high bandwidth variability, low power and resources, security risks, wide variety of devices with different capabilities and to fit more functionality into single, smaller devices. There are many famed cryptography procedures used for mobile security [36][37]. The public-key algorithms like RSA and ECC are taken in this thesis. They are used for digital data security in a great extend. RSA is a procedure of computational simplicity whereas ECC provides greater security. Large prime numbers are used as security keys in these methods.

ECC gets popularity due to its shorter key length which produces same security as in RSA with larger key length. It is noted that if a cryptographic algorithm takes more time in performing operational time (where the operational time includes both encryption and decryption) which causes customer impatience and dissatisfaction. Thus, to minimize the operational time in RSA and ECC, AC is incorporated in performing $x^e mod\ n$ of RSA and k[P] of ECC where the AC is generated using PSO, SSO and BFO.

3.4.1 RSA

RSA is one of the most used asymmetric cryptographic algorithms. It was developed in 1977 by Ronald-Alan Rivest, Adi Shamir, and Leonard Adleman. RSA is a popular algorithm because it is a simple, easy to understand and to implement. The only disadvantage is, it works slower than symmetric block ciphers. Most systems use RSA for the generation of digital signature and distribution of the symmetric keys. Recently key lengths of 4096 bits have been used in most system. Developers have organized into RSA Laboratories to define the syntax of different structures related to public-key cryptography and private-keys. These guidelines are de facto standards and are known as Public-Key Cryptography Standards (PKCS). RSA algorithm consists of three components viz., key generation, performing encryption and performing decryption and they are shown in algorithms 3.1, 3.2 and 3.3 respectively. It is noted that in RSA, Alice (A) is receiver and Bob (B) is sender.

Algorithm 3.1: RSA Cryptosystem: RSA-Key generation

Each entity creates an RSA public-key and a corresponding private-key.

Each entity A should do the following:

- ✓ Generate two large random (and distinct) primes p and q, each roughly the same size.
- ✓ Compute $n = p$ x q and $\phi(n) = (p - 1)$ x $(q - 1)$.
- ✓ Select a random integer $e, 1 < e < \phi(n)$ such that $gcd(e, \phi(n)) = 1$.
- ✓ Use the extended Euclidean algorithm to compute the unique integer $d, 1 < d < \phi(n)$, such that $e * d \equiv 1 \ (mod\ \varphi(n))$.
- ✓ A's public-key is (n, e).
- ✓ A's private-key is (n, d).

Algorithm 3.2: RSA- Encryption

B encrypts a message m for A, which A decrypts.

***Encryption**: B* should do the following:

- ✓ Obtain $A's$ authentic public-key (n, e).
- ✓ Represent the message m as an integer m in the interval $\{0, 1, 2, \ldots . n - 1\}$.
- ✓ Compute the ciphertext $c = m^e mod\ n$
- ✓ Send the ciphertext c to A.

Algorithm 3.3: *RSA -Decryption*

To recover plaintext m from c, A should do the following:

- ✓ Use the private-key d to recover $m = c^d\ mod\ n$.

- **RSA Cryptosystem – An Example**

Let Entity A chooses the two primes $p = 2357, q = 2551$. Then n is computed as $n = pq = 6012707$ and $\phi(n) = (p-1)(q-1) = 6007800$. Let A chooses $e = 3674911$ because $gcd(3674911, 6007800) = 1$. Using Extended Euclidean algorithm d is computed as $d = 422191$. Now, $A's$ public-key is the pair $(n = 6012707; e = 3674911)$, while $A's$ private-key is $(n = 6012707; d = 422191)$. Suppose B wants to send the message $m = 5234673$ to A. Then, B uses an algorithm for modular exponentiation to compute

$c = m^e mod\ n = 5234673^{3674911}\ mod\ 6012707 = 3650502$ and sends this to A. After receiving c, c is decrypted by A by computing $c^d\ mod\ n = 3650502^{422191}\ mod\ 6012707 = 5234673 = m$.

3.5 Mathematical Preliminaries of ECC

The following mathematical preliminaries are required to understand the concept of EC and ECC.

Theorem 3.5.1: Primitive Root

Let p be a prime number. Then there exists an element $G \in F_p^*$ whose powers give every element of F_p^*, (i.e.) $F_p^* = \{1, G, G^2, G^3, \dots, G^{p-2}\}$ ... (3.4)

Elements with this property are called primitive roots of F_p or generators of F_p^*. They are the elements of F_p^* having order $p - 1$.

Theorem 3.5.2: Euler's Criterion

Let p be an odd prime and a be an integer. Then

$$a^{\frac{(p-1)}{2}} \equiv 1\ mod\ p \quad \text{... (3.5)}$$

3.5.3 Definition (Discrete Logarithm Problem)

Let G be the primitive root for finite field F_p and let h be a nonzero element of F_p.

The DLP is the problem of finding an exponent (x) such that

$$g^x \equiv h\ (mod\ p) \quad \text{... (3.6)}$$

The number x is called as DLP of h.

3.5.4 Definition (Quadratic Residue)

Let p be an odd prime and $\gcd\ (a, p) = 1$. If the quadratic congruence $x^2 \equiv a\ mod\ p$ has a solution, then a is said to be quadratic residue of p i.e., Q_R(p), otherwise a is called quadratic- non residue of p i.e., Q_{NR}(p).

3.6 Need for ECC

ECC is an approach to public-key cryptography based on the algebraic structure of EC over finite fields. It allows smaller keys compared to non – EC cryptography to provide equivalent security. ECC, an alternative technique to RSA, is a powerful cryptography approach. It generates the security between key pairs for public-key encryption by using the mathematics of EC. ECC creates keys that are more difficult and mathematically crack. For this reason, it also makes sense to adopt ECC to maintain high levels of both performance and security. ECC provides the same security as RSA, with the fewer number of bits i.e., keys that are 2048 bits or longer used in RSA makes the process slow and it also means that key size is important. Size is a serious advantage of ECC, because it translates into more power for smaller, mobile devices. In devices with limited memory and computing power, ECC could become alternatives to other public-key systems. In a world where mobile devices must do more and more cryptography with less computational power, ECC offers high security with faster, shorter keys compared to RSA.

3.7 Concepts of ECC

It is noted that the security of RSA algorithm depends on large key values viz., 2096 or 4096 bits. To provide the same security, ECC is normally used with smaller key length i.e., a 256 bits ECC is considered to be equivalent to 3072 bits RSA. The concept of ECC was developed by two mathematicians Neal Koblitz and V.S. Miller independently [99-104]. It is based on algebraic structure of EC over the finite field F and has the two variables. The cubic EC is of the form

$$y^2 \equiv x^3 + ax + b \bmod p \quad \ldots (3.7)$$

together with point at infinity O. The point of O is similar to the number 0 as in normal addition where a and b define the shape of the curve and p is a modulo p prime for fixing the range of the curve. Also a and b are selected in such a way that it must satisfy the Weistrass equation

$$4a^3 + 27b^2 \neq 0$$

EC over the finite field F_p is the set of points $(x, y) \in F_p \times F_p$ satisfying the Weiestrass equation. ECC is frequently discussed in the context of the RSA cryptographic algorithm. RSA achieves one-way encryption of things like emails, data, and software using prime factorization. EC is shortly denoted as $Ep\ (a, b)$.

3.7.1 Generation of EC Points

To generate the points of EC, consider $E: y^2 = x^3 + ax + b$ with a and b value is taken as 1 and prime $p = 31$. To generate the points of $E_{31}(1,1)$, first find $Q_R(31)$. Then, generate the points for EC. Table 3.1 shows the EC points for the curve $E_{31}(1, 1)$

Table 3.1: Generations of Points for $E_{31}(1, 1)$

x	x^3+x+1	$b=x^3+x+1 \bmod 31$	$b^{15} \bmod 31$	$y^2 \epsilon Q_{31}$	$E_{31}(1,1)$ Points
0	1	1	1	Y	(0,1) (0,30)
1	3	3	30	N	-
2	11	11	30	N	-
3	31	0	0	N	-
4	69	7	1	Y	(4,10),(4,21)
5	131	7	1	Y	(5,10),(5,21)
6	223	6	30	N	-
7	351	10	1	Y	(7,14),(7,17)
8	521	25	1	Y	(8,5),(8,26)
9	739	26	25	N	-
10	1011	19	1	Y	(10,9),(10,22)
11	1343	10	1	Y	(11,14),(11,17)

12	1741	5	1	Y	(12,6),(12,25)
13	2211	10	1	Y	(13,14),(13,17)
14	2759	0	0	N	--
15	3391	12	30	N	-
16	4113	21	25	N	-
17	4931	2	1	Y	(17,8),(17,23)
18	5851	23	14	N	-
19	6879	28	1	Y	(19,11),(19,20)
20	8021	23	14	N	-
21	9283	14	1	Y	(21,13),(21,18)
22	10671	7	1	Y	(22,10),(22,21)
23	12191	8	1	Y	(23,15),(23,16)
24	13849	23	14	N	-
25	15651	27	24	N	-
26	17603	26	25	N	-
27	19711	26	25	N	-
28	21981	2	1	Y	(28,8),(28,23)
29	24419	22	30	N	-
30	27031	30	16	N	-

The said points are plotted and the $E_{31}(1,1)$ is obtained. It is shown in fig. 3.2.

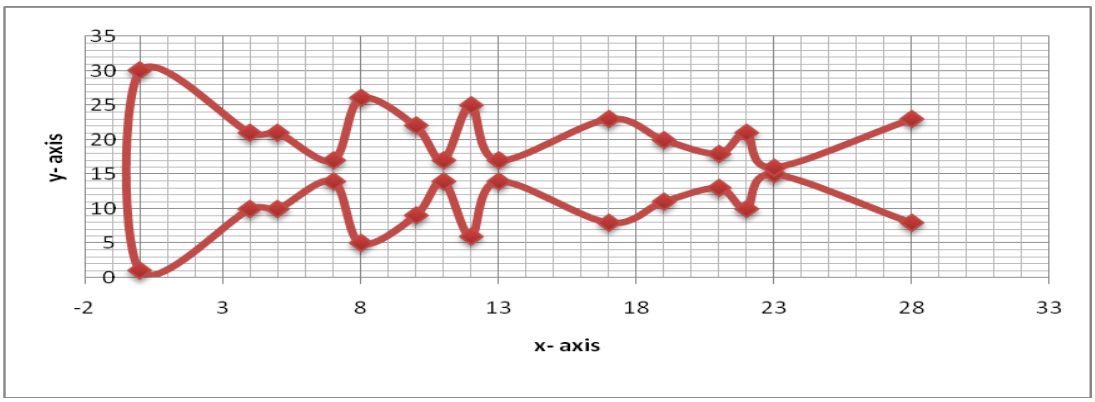


Fig. 3.2: Graph Showing $E_{31}(1,1)$

3.7.2 Elliptic Curve Arithmetic

- **Addition of Points**

Let $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ be the two points on EC. The sum of two points $p_3 = p_1 + p_2$, where $p_3 = (x_3, y_3)$ is calculated as follows.

Case 1:

If $p_1 \neq p_2$, then

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{... (3.8)}$$

$$x_3 = \lambda^2 - x_1 - x_2 \quad \text{... (3.9)}$$

$$y_3 = (x_1 - x_3)\lambda - y_1 \quad \text{... (3.10)}$$

Case 2:

If $p_1 = p_2$, then

$$\lambda = \frac{3x_1^2 + a}{2y_1} \quad \text{... (3.11)}$$

$$x_3 = \lambda^2 - 2x_1 \quad \text{... (3. 12)}$$

$$y_3 = (x_1 - x_3)\lambda - y_1 \quad \text{... (3.13)}$$

- **Scalar point Multiplication**

The central operation of ECC is the scalar point multiplication $[k]P$, where k is an integer and P is a point on EC. The $[k]P$ is the result of adding P to itself k times, where $1 \leq k \leq ord(P)$.

3.8 Embedding the Plaintext

In order to embed the plaintext M into $E_p(a, b)$ points, the value of k should be $30 \leq k \leq 50$. Let $m_i \in M$, is the individual characters of M. To embed m_i, it must satisfy the condition $(m_i + 1)\, k < p$ of the $E_p(a, b)$. Once the condition is checked using $(mk) + j$. where j should be $0 < j < 30$ embed m_i into $E_p(a, b)$ points and they are used for encryption and decryption with any one of the cryptosystems viz., RSA, ECC, ElGamal etc.

For example, let M is taken as "$KANNANBABA$" and EC is taken as $E_{539039}(17,7)$. The points for $E_{539039}(17,7)$ are generated and they are shown in appendix $(A.1)$. To

embed M, let $k = 30$ or worst case $k = 50$, then $(m + 1)30 < 539039$. Suppose, ASCII encoding is used for each $m_i \in M$ then $ASC(m_i)$ = {75,65,78,78,65,08,66,65,66, 65}. To embed m_i i.e., Em_i, $i = 1,2,\ldots,11$ in E, it satisfies $(m_i + 1)30 < 539039$. Further, $m_i(k) + j$, $0 \leq j < 30$. For example, to embed $m_1 =$ "K", then $ASC(m_1) = 75$. From EC points, $x_1 = 2252$ when $j = 2$. Thus, $Em_1(k)$= $(2252, 226996)$. Other m_i, $i = 2,\ldots,11$ are embedded in EC points in this manner and they are shown in table 3.2.

Table 3.2: Embedding $\mathbf{M}$ into $E_{539039}(17,7)$

i	M_I	ASC(m_i)	(ASC(m_i)+1)30<p	$x_i=m_i(30)+j$	E_{m_i}
1	K	75	2280	2250	(2252,226996)
2	A	65	1980	1950	(1950, 246296)
3	N	78	2370	2340	(2340,153325)
4	N	78	2370	2340	(2340,153325)
5	A	65	1980	1950	(1950,292743)
6	N	78	2370	2340	(2340,385714)
7	blank	08	270	240	(240,102442)
8	B	66	2010	1980	(1981,74914)
9	A	65	1980	1950	(1950,246296)
10	B	66	2010	1980	(1981,48955)
11	A	65	1980	1950	(1950,282743)

3.9 ElGamal Public-key Cryptosystem with EC

In order to understand the EC with any cryptosystem, ElGamal proposed a public-key cryptosystem which is based on the Discrete Logarithm Problem (DLP) in $(Z_p^*, +)$. This system is presented in ElGamal Public-key Cryptosystem in Z_p^* . Let p be a

prime such that the DLP in $(Z_p^*, +)$ is reliable, and let $\propto \in Z_p^*$ be a primitive element. Let $P = Z_p^*$, $ç = Z_p^* * Z_p^*$ and define

$$K = \{(p, \alpha, a, \beta): \beta \equiv \alpha^a (mod\ p) \quad \ldots (3.14)$$

The values of $p, \propto$ and β are the public-key, and α is the private-key. For $K = (p, \propto, a, \beta)$, and for a (secret) random number $k \epsilon Z_{p-1}$, define

$$e_k(x.k) = (k\alpha, x + k\beta) = (y_1, y_2) \quad \ldots (3.15)$$

where $y_1 = a^k\ mod\ p$ and $y_2 = x\beta^k\ mod\ p$ … (3.16)

For y_1 and $y_2 \epsilon Z_p^*$, define

$$d_k(y_1, y_2) = y_2 (y_1{}^a)^{-1}\ mod\ p \quad \ldots (3.17)$$

3.9.1 ElGamal Encryption with EC - An Example

Let the primitive element $\alpha = (1950, 246296)$ and $a = 7$. Now,

$$\beta = 7\alpha = 7(1950, 246296) = (533025, 457088)$$

$$e_k(x.k) = (k\alpha, x + k\beta) = (y_1, y_2) \quad \ldots (3.18)$$

Let $k = 3$ and $x = m_i \epsilon E, y_1\ and\ y_2$ are ciphertexts and the decryption operation is

$$d_k(y_1, y_2) = y_2 - \beta \quad \ldots (3.19)$$

Suppose, Alice wishes to send the message M (here x) , say *"KANNAN BABA"* to Bob. Now, Alice encrypts the character one by one. Now, m_1 = K, m_2 = "A", …, m_{10} = "A". To encrypt $x_1 = m_1 = K$, from table 3.2, it is embedded into EC as $Em_1 = (2252, 226996)$. Using eqn. (3.15) and eqn. (3.16) y_1 and y_2 are computed as

$$y_1 = 3(1950, 246296) = (397955, 288822)$$

y_2 = (2252, 226996) + 3(533025, 57088)

= (2252, 226996) + (77065, 118651) = (80907, 94455)

To decrypt (y_1, y_2), Bob must do as

$$d_k = (y_1, y_2) = y_2 - 7y_1 \quad \ldots (3.20)$$

$$= (80907, 94455) - 7((397955, 288822)$$

$$= (80907, 94455) - (345943, 368384)$$

$$= (80907, 94455) + (345943, 170655)$$

$$= (2252, 226996) = Em_1 = K$$

Similar computations can also be performed for other characters too to encrypt and decrypt them using ElGamal with EC.

3.10 Diffie Helman Key Exchange Protocol with ECC

1. A selects an integer n_A less than n. This is A's private-key. A then generates a public-key $P_A = n_A \; X \; G$; the public-key is a point $Eq(a, b)$.

2. B similarly selects a private-key n_B and computes a public-key P_B.

3. A generates the secret key $k = n_A \; X \; P_B$. B generates the secret key $k = n_B \; X \; P_A$.

- **Global Public Elements**

  $Eq(a, b)$ EC parameters a, b and q, where q is a prime or an integer of the form 2^m.

  G point on EC whose order is large value n

- **User A Key Generation**

  Select private n_A $n_A < n$

  Calculate public P_A $P_A = n_A \; X \; G$

- **User B Key Generation**

  Select private n_B $n_B < n$

  Calculate public P_A $P_B = n_B \; X \; G$

- **Calculation of Secret Key by User A**

  $k = n_A \; X \; P_B$

- **Calculation of Secret Key by User B**

  $k = n_B \; X \; P_A$

The two calculations in step 3 produce the same result because

$$n_A \; X \; P_B = \; n_A \; X \; (n_B \; X \; G) = n_B X (n_A \; X \; G \;) = n_B \; X \; P_A$$

3.11 Mobile Operating Systems

In order to functioning the mobile devices MOS are normally used. This section describes an overview of MOS.

Smart phones are not only used to make calls but also used for other operations like video calls, multimedia messages, take pictures, play media files, browse World Wide Web (WWW), run web applications i.e., multiple tasks run on the device. Powerful OS has become an essential part and available in various forms depends on the sophistications of the device level [53]. It can do two things viz., (i) managing the resources (camera, speaker, keyboard, and screen) (ii) providing different interfaces (user of the device and also several devices with networks) .

3.12 Constraints of MOS

The special constraints of MOS are:

- ✓ Severely limited energy stored in a tiny battery
- ✓ Limited memory space
- ✓ Limited screen size
- ✓ Miniature keyboard
- ✓ Limited processing power
- ✓ Limited battery power
- ✓ Limited and fluctuating bandwidth of the wireless medium and
- ✓ Real-time data streaming etc.

The MOS is designed to run on mobile devices such as mobile phones, smartphones, tablet and other handheld devices. Even though, different types of MOS include Apple

iOS, Google Android, BlackBerry OS, Nokia's Symbian and Microsoft's Windows Phone OS exist. In this thesis, Android OS and Windows OS will be taken for the test cases because they are based on Linux OS and Microsoft OS respectively.

3.13 Android and Window OS Emulators

In this thesis, to implement the proposed algorithms only Android (A) and Windows (W) emulators are taken. Android is one of the popular OSs developed by Google. It is based on the Linux environment. It is a free open source software. Samsung, HTC, Micromax, Motorola and many other top manufacturers are using android in their devices. Windows OS is proprietary mobile OS developed by Microsoft for Smartphone. It was very popular among people who were used to it. Windows OS provide colourful and user-friendly interface so currently in demand all over the world. Fig. 3.3 shows the GUI of Android and Windows emulators.

Fig. 3.3: GUI - Android/ Windows Emulator Launching

3.14 Need for Security

Mobile security or mobile device security is becoming more notable within modern technology. The personal information is that it can be found on smart phone nowadays. More users are using smart phones to communicate, organize their

schedule and their lives. There is a need for confidentiality, integrity, authenticity, authorization and non-repudiation in database security. Among the additional challenges for multi-location database is the constant mobility of its users and the portability of handheld devices and wireless links. Some security issues due to mobile users, hackers and viruses are vulnerable nowadays. In order to secure database, authentication mechanism is provided, control access scheme and strong encryption technique must be implemented [57].

Public-key algorithms like RSA and ECC have become much more widely utilised than symmetric key systems. ECC gets creditability because the same security level is produced by the shorter key length. This algorithm's lack of performance causes customer discontent. Because it requires more operational time (where operational time includes time taken for encryption and decryption) which leads to customer's impatience and dissatisfaction.

In RSA, for encryption and decryption, exponentiation operation is involved which takes more time. To reduce the time, exponentiation operation is performed by RMs. For example, to compute x^5, the proposed RMs are $x * x * x * x * x$. In general, for performing x^e, $(e-1)$ multiplications are required. Similarly, in ECC, to perform $k[P]$, $(k-1)$ RAs are required which take somewhat less time when it is compared with classical multiplications used in ECC.

3.15 Experimental Set up

Two different types of mobile emulators A and W are used in this work. The T-Engine A emulator and W emulator 6.1.4 are used to set up the A and W - OS mobile infrastructure respectively. The User Interface (UI) is designed using Visual C++ to upload files for A and W emulators. UI and emulator executions are carried out on a

Windows 8.1 64-bits Intel i5 2.4 GHz processor-4GB RAM computer. To load and execute various encryption algorithms, each emulator is individually used. In order to test the proposed algorithms, the experiments are not stopped until either 500 iterations or it reaches optimal AC and the maximum swarm size is taken as 180.

3.16 Parameters Taken in the Work

There are five parameters mainly used in mobile emulators viz., Encryption Time (ET), Decryption Time (DT), Encryption Power (EP), Decryption Power (DP) and Security (SE). ET and DT are the time taken for converting M into C and vice versa respectively. It includes the time taken for all arithmetic computation of key generation and encoding of M. Without compromising the security, a good cryptography algorithm should be able to encrypt the data more quickly. In measuring the quality of a cryptography algoithm, this DT parameter gets equal priority to the ET. Shorter DT refers to the quality of cryptography algorithm is high.

The energy consumed by encoding and encrypting M into C and C into M is called EP and vice versa EP and DP respectively. Mobile devices are battery-powered devices that enable greater mobility with careful consumption of power. Without compromising security strength, a successful cryptography algorithm should be capable of processing with acceptable power consumption.

Producing the security is an another target of cryptography algorithms. Without using a lot of computing resources, the best cryptography algorithm can provide high security. The cryptography algorithm should operate with greater power-awareness, especially in the architecture of mobile devices. With the assistance of key sizes, key revocations and procedural complexity, security strength can be measured

mathematically. Security strength can be calculated using certain crypt-analysis methods that can use different attacks to attempt to breach security. The ultimate goal of an ideal cryptography algorithm is to achieve higher security with less operational time and energy consumption. The security level produced by the existing and the proposed algorithms are measured by All Block Ciphers (ABC) Universal Hackman tool which uses dictionary attack.

3.17 Results and Discussion

The regular RSA and ECC, RM-RSA and RA-ECC are implemented using VC++. The results obtained are tabulated. Tables 3.3 to 3.12 show the time taken for ET, DT, EP, DP and SE with and without using RM and RA in RSA and ECC respectively with A and W emulators. Their corresponding graphical representation of said tables are shown in fig. 3.4 to 3.13.

Table 3.3: Encryption Time (mS) using Android Emulator

File Size (MB)	ET-RSA-A	ET-ECC-A	ET-RM-RSA-A	ET-RA- ECC-A
1	1660	2447	1227	1820
2	3237	4790	2377	3555
4	6494	9553	4788	7173
8	13689	20179	10077	15130
16	27426	40415	20199	30326
Total	52506	77384	38668	58004
Avg.	10501.2	15476.8	7733.6	11600.8

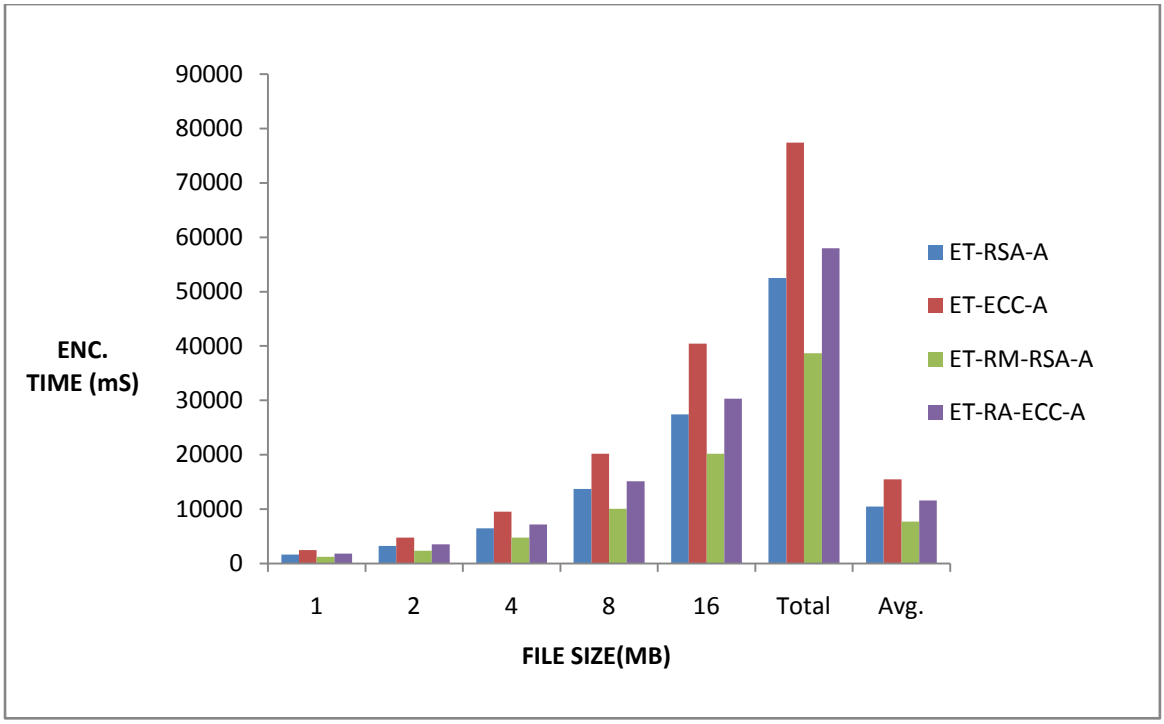


Fig. 3.4: Graph Showing Encryption Time using Android Emulator

- ET-RSA-A is 1.474 times faster than ET-ECC-A
- ET-RM-RSA-A is 1.500 times faster than ET-RA- ECC-A
- ET-RM-RSA-A is 1.358 times faster than ET-RSA-A
- ET-RA- ECC-A is 1.334 times faster than ET-ECC-A

Table 3.4: Decryption Time (mS) using Android Emulator

File Size (MB)	DT-RSA-A	DT-ECC-A	DT-RM-RSA-A	DT-RA- ECC-A
1	1616	2330	1205	1789
2	3193	4519	2352	3519
4	6490	9209	4776	7171
8	13645	19375	10066	15082
16	27399	38936	20193	30294
Total	52343	74369	38592	57855
Avg.	10468.6	14873.8	7718.4	11571

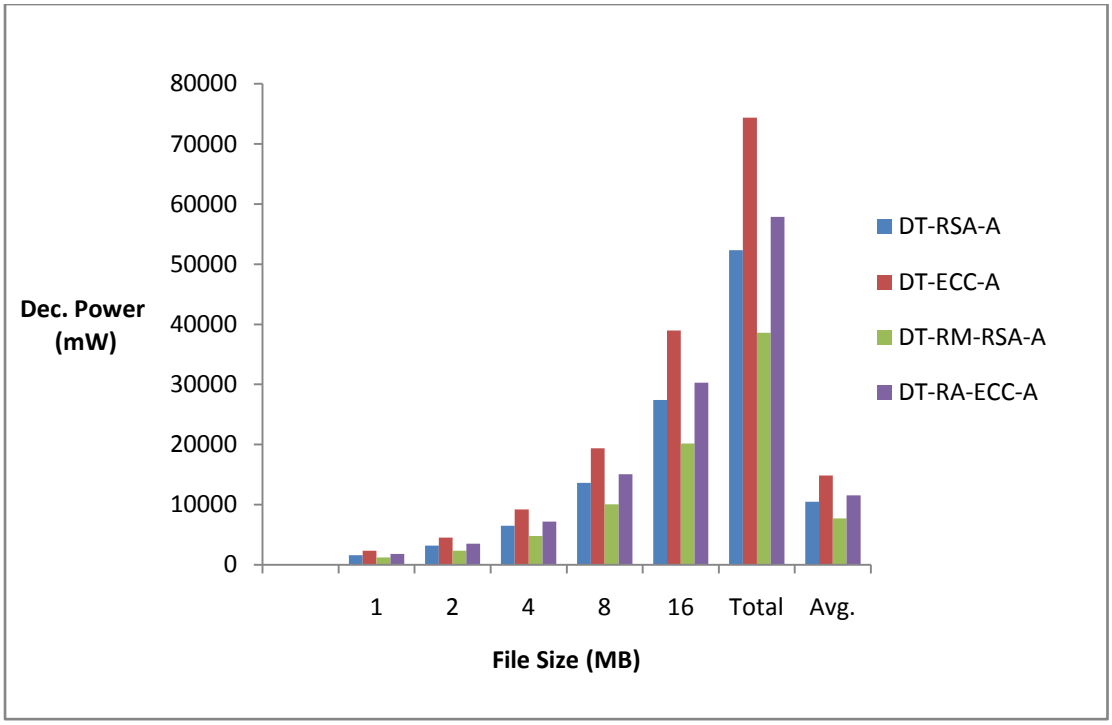


Fig. 3.5: Graph Showing Decryption Time using Android Emulator

- DT-RSA-A is 1.421 times faster than DT-ECC-A
- DT-RM-RSA-A is 1.499 times faster than DT-RA- ECC-A
- DT-RM-RSA-A is 1.356 times faster than DT-RSA-A
- DT-RA- ECC-A is 1.285 times faster than DT-ECC-A

Table 3.5: Encryption Power (mW) using Android Emulator

File Size (MB)	EP-RSA-A	EP-ECC-A	EP-RM-RSA-A	EP-RA- ECC-A
1	554	817	421	613
2	1102	1621	806	1197
4	2171	3208	1612	2410
8	4569	6731	3364	5042
16	9156	13510	6735	10134
Total	17552	25887	12938	19396
Avg.	3510.4	5177.4	2587.6	3879.2

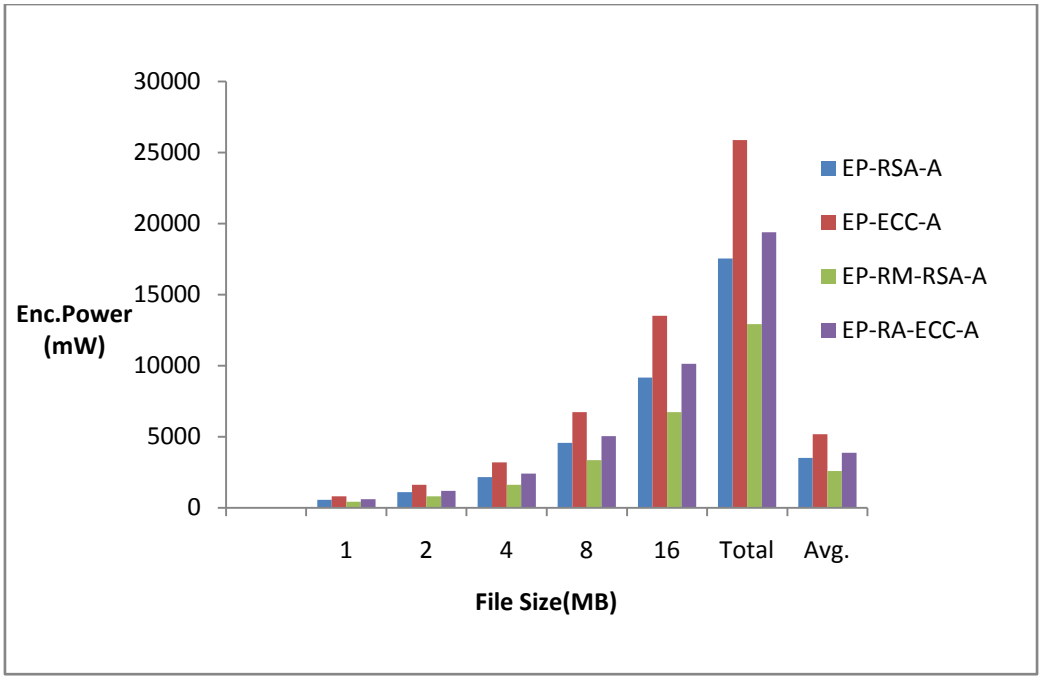


Fig. 3.6: Graph Showing Encryption Power (mW) using Android Emulator

- EP-RSA-A is 1.475 times less than EP-ECC-A
- EP-RM-RSA-A is 1.499 times less than EP-RA- ECC-A
- EP-RM-RSA-A is 1.357 times less than EP-RSA-A
- EP-RA- ECC-A is 1.335 times less than EP-ECC-A

Table 3.6: Decryption Power using Android Emulator

File Size (MB)	DP-RSA-A	DP-ECC-A	DP-RM-RSA-A	DP-RA- ECC-A
1	562	815	411	602
2	1081	1506	802	1178
4	2175	3072	1608	2417
8	4548	6463	3357	5035
16	9148	12995	6743	10109
Total	17514	24851	12921	19341
Avg.	3502.8	4970.2	2584.2	3868.2

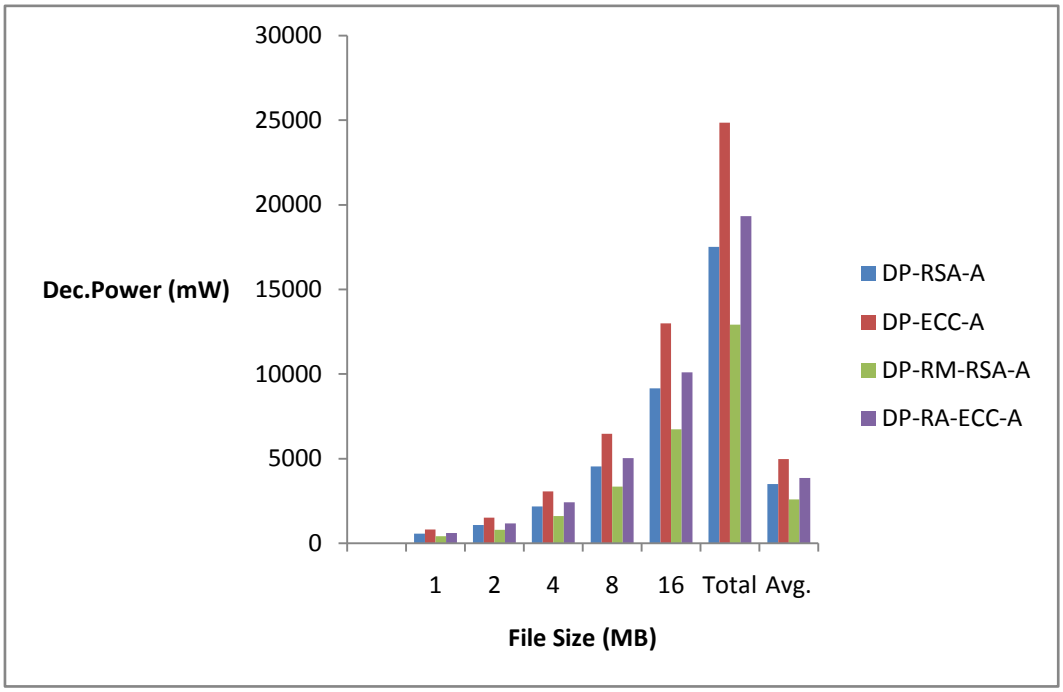


Fig. 3.7: Graph Showing Decryption Power using Android Emulator

- DP-RSA-A is 1.419 times less than DP-ECC-A
- DP-RM-RSA-A is 1.497 times less than DP-RA- ECC-A
- DP-RM-RSA-A is 1.355 times less than DP-RSA-A
- DP-RA- ECC-A is 1.285 times less than DP-ECC-A

Table 3.7: Security (%) using Android Emulator

File Size (MB)	SE-RSA-A	SE-ECC-A	SE-RM-RSA-A	SE-RA- ECC-A
1	89	93	92	94
2	88	89	89	92
4	87	88	88	90
8	85	88	87	90
16	85	86	87	89
Avg.	86.8	88.8	88.6	91

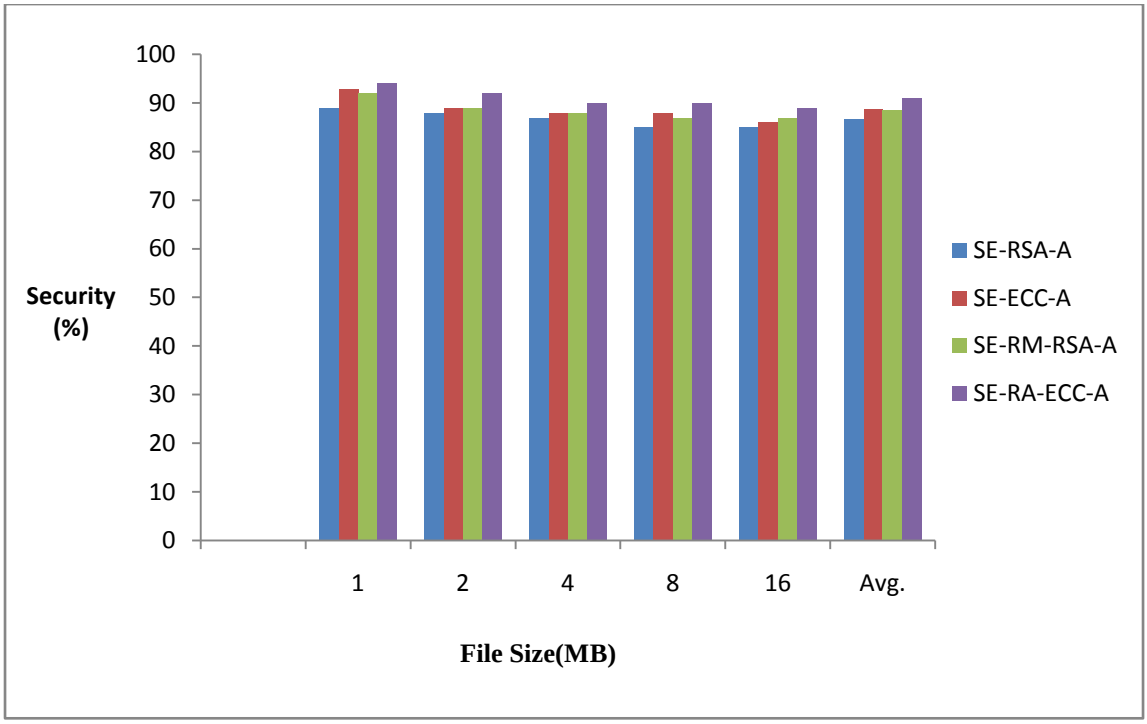


Fig. 3.8: Graph Showing Security using Android Emulator

- SE-ECC-A is 1.023 times more than SE-RSA-A
- SE-RA- ECC-A is 1.027 times more than SE-RM-RSA-A
- SE-RM-RSA-A is 1.021 times more than SE-RSA-A
- SE-RA- ECC-A is 1.025 times more than SE-ECC-A

Table 3.8: Encryption Time (mS) using Windows Emulator

File Size (MB)	ET-RSA-W	ET-ECC-W	ET-RM-RSA-W	ET-RA- ECC-W
1	1654	2441	1204	1728
2	3233	4748	2378	3362
4	6490	9559	4775	6753
8	13670	20174	10082	14252
16	27432	40443	20214	28603
Total	52479	77365	38653	54698
Avg.	10495.8	15473	7730.6	10939.6

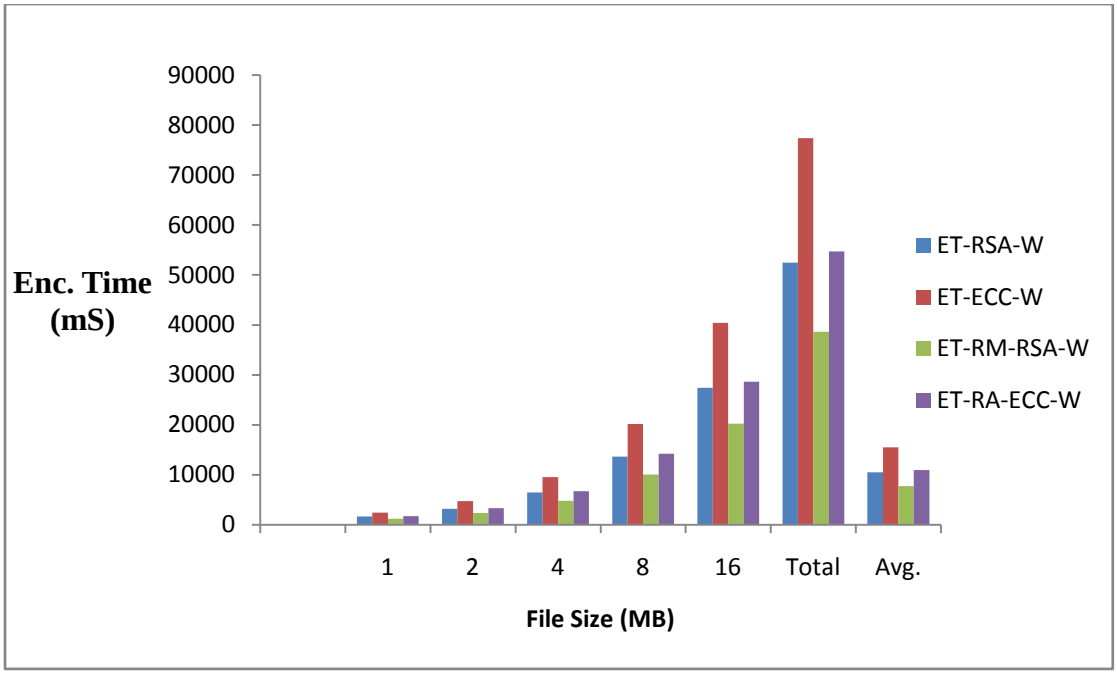


Fig. 3.9: Graph Showing Encryption Time using Windows Emulator

- ET-RSA-W is 1.474 times faster than ET-ECC-W
- ET-RM-RSA-W is 1.415 times faster than ET-RA-ECC-W
- ET-RM-RSA-W is 1.358 times faster than ET-RSA-W
- ET-RA- ECC-W is 1.414 times faster than ET-ECC-W

Table 3.9: Decryption Time using (mS) Windows Emulator

File Size (MB)	DT-RSA-W	DT-ECC-W	DT-RM-RSA-W	DT-RA- ECC-W
1	1616	2322	1205	1708
2	3184	4530	2366	3336
4	6496	9212	4778	6755
8	13657	19396	10050	14240
16	27401	38958	20210	28571
Total	52354	74418	38609	54610
Avg.	10470.8	14883.6	7721.8	10922

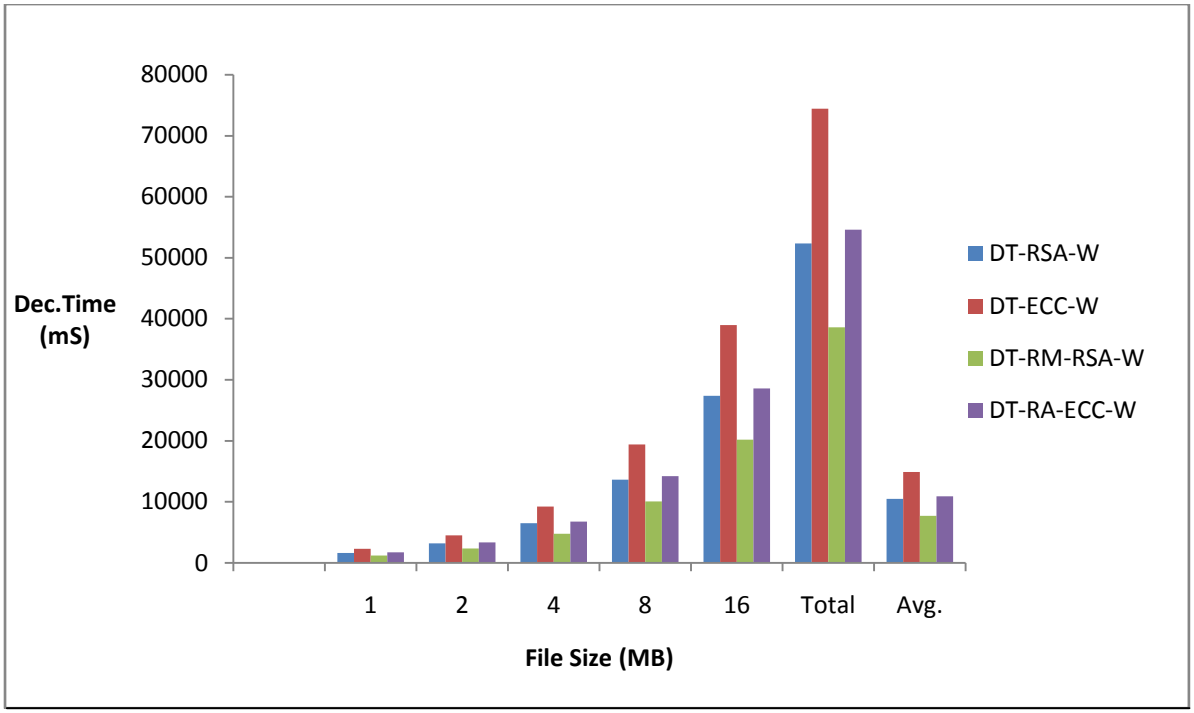


Fig. 3.10: Graph Showing Decryption Time using Windows Emulator

- DT-RSA-W is 1.421 times faster than DT-ECC-W
- DT-RM-RSA-W is 1.414 times faster than DT-RA-ECC-W
- DT-RM-RSA-W is 1.356 times faster than DT-RSA-W
- DT-RA-ECC-W is 1.363 times faster than DT-ECC-W

Table 3.10: Encryption Power (mW) using Windows Emulator

File Size (MB)	EP-RSA-W	EP-ECC-W	EP-RM-RSA-W	EP-RA- ECC-W
1	571	840	421	592
2	1100	1582	796	1146
4	2179	3228	1604	2263
8	4577	6750	3361	4765
16	9165	13496	6738	9559
Total	17592	25896	12920	18325
Avg.	3518.4	5179.2	2584	3665

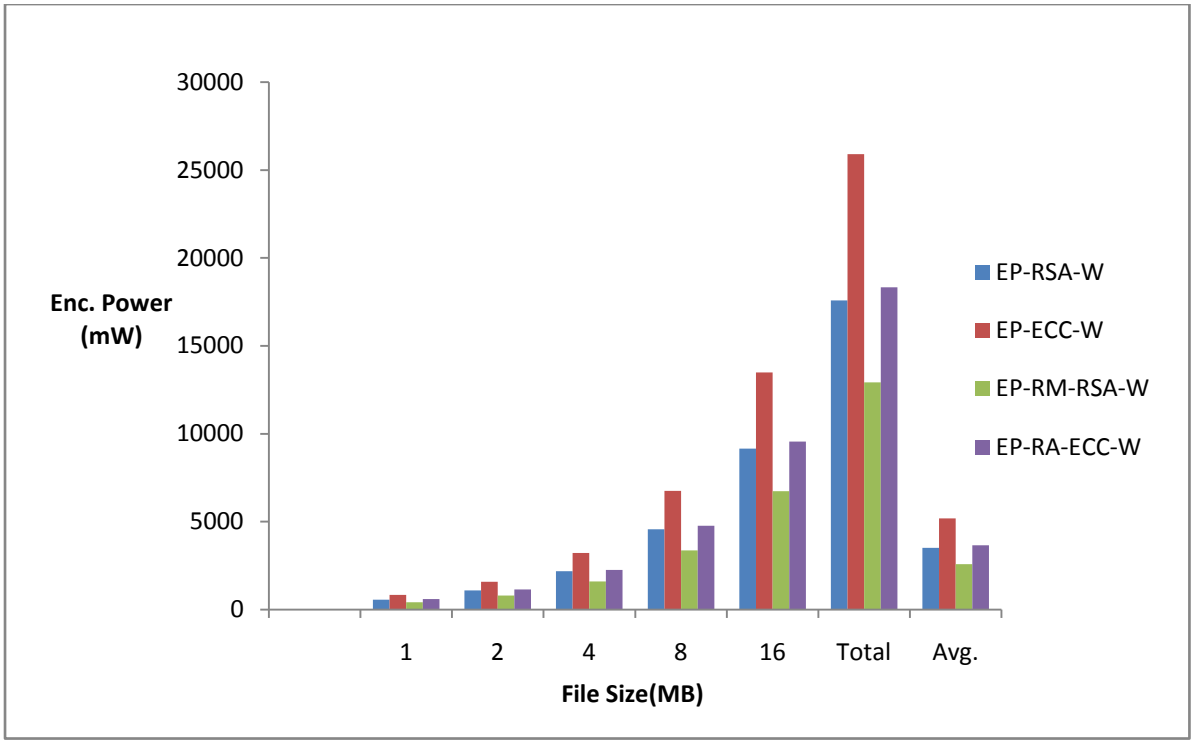


Fig. 3.11: Graph Showing Encryption Power using Windows Emulator

- EP-RSA-W is 1.362 times less than EP-ECC-W
- EP-RM-RSA-W is 1.413 times less than EP-RA- ECC-W
- EP-RM-RSA-W is 1.362 times less than EP-RSA-W
- EP-RA- ECC-W is 1.413 times less than EP-ECC-W

Table 3.11: Decryption Power (mW) using Windows Emulator

File Size (MB)	DP-RSA-W	DP-ECC-W	DP-RM-RSA-W	DP-RA- ECC-W
1	541	785	415	584
2	1086	1536	805	1140
4	2192	3088	1604	2275
8	4579	6496	3368	4761
16	9133	12997	6742	9529
Total	17531	24902	12934	18289
Avg.	3506.2	4980.4	2586.8	3657.8

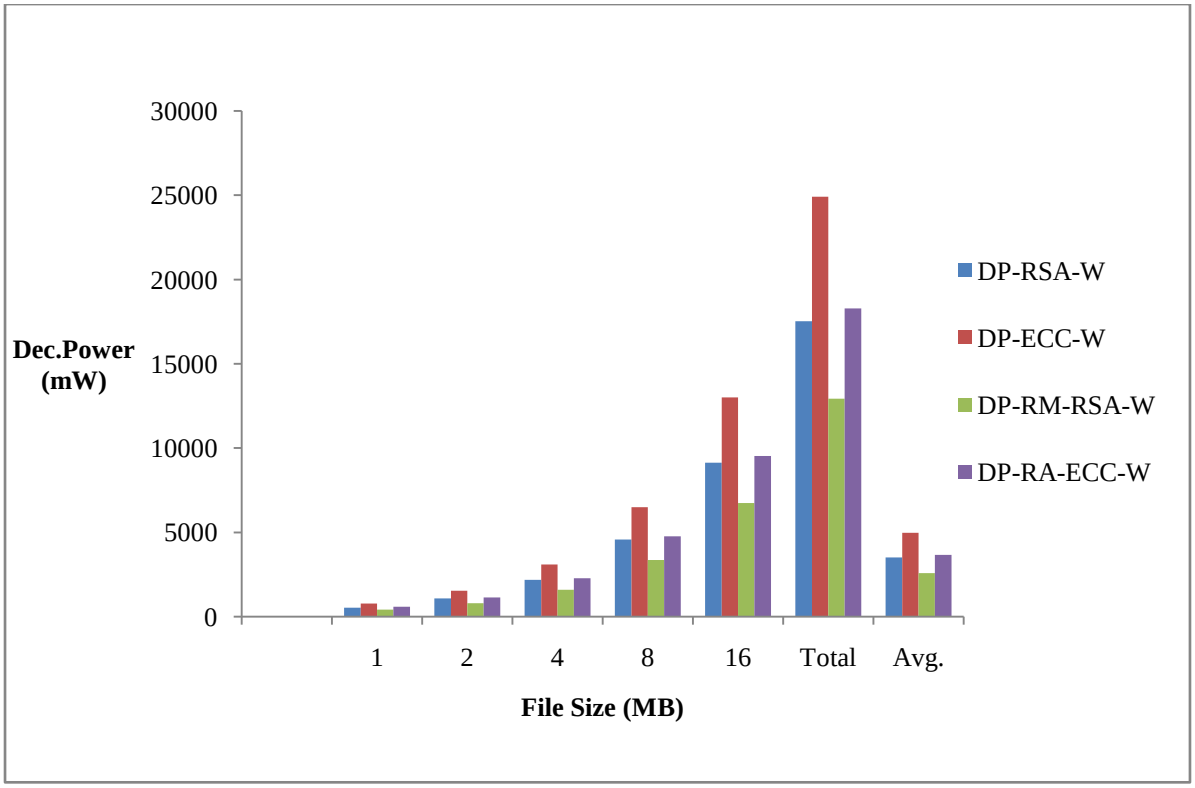


Fig. 3.12: Graph Showing Decryption Power using Windows Emulator

- DP-RSA-W is 1.420 times less than DP-ECC-W
- DP-RM-RSA-W is 1.414 times less than DP-RA- ECC-W
- DP-RM-RSA-W is 1.355 times less than DP-RSA-W
- DP-RA- ECC-W is 1.362 times less than DP-ECC-W

Table 3.12: Security (%) using Windows Emulator

File Size (MB)	SE-RSA-W	SE-ECC-W	SE-RM-RSA-W	SE-RA- ECC-W
1	91	92	92	94
2	88	89	89	92
4	86	89	88	91
8	86	88	88	89
16	85	87	87	89
Avg.	87.2	89	88.8	91

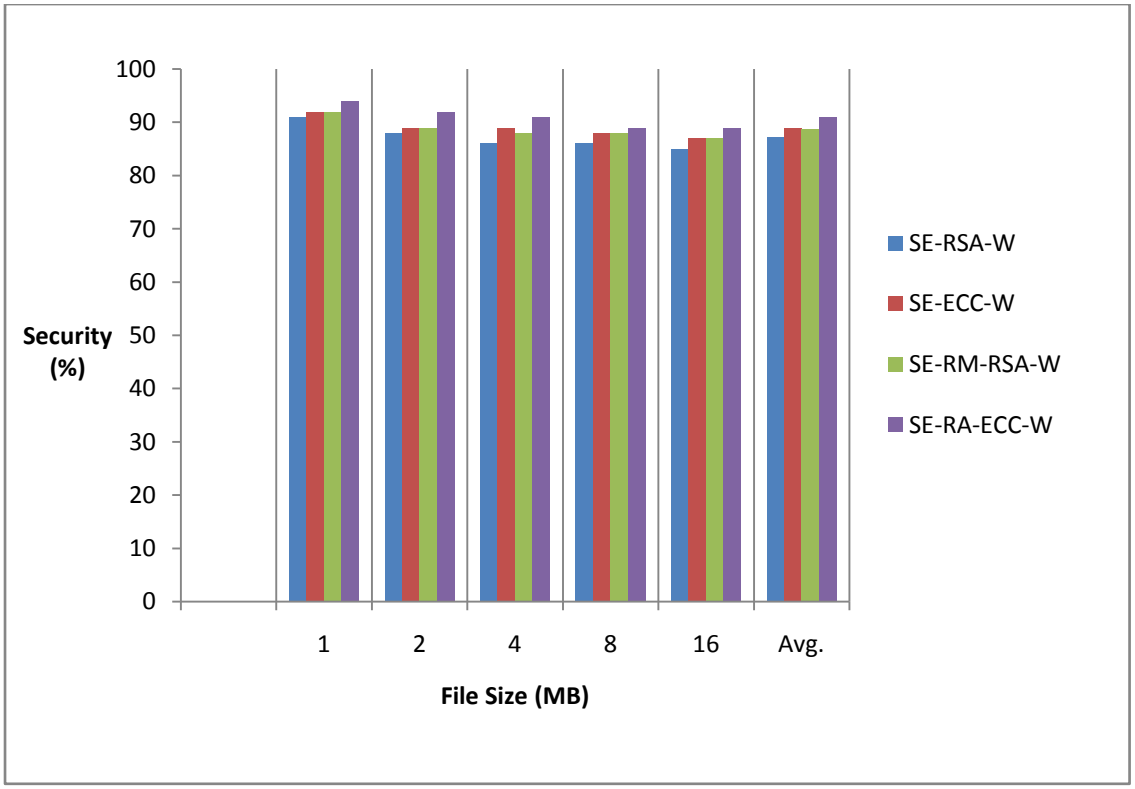


Fig. 3.13: Graph Showing Security using Windows Emulator

- SE-ECC-W is 1.021 times more than SE-RSA-W
- SE-RA- ECC-W is 1.025 times more than SE-RM-RSA-W
- SE-RA- ECC-W is 1.022 times more than SE-ECC-W
- SE-RM-RSA-W is 1.018 times more than SE-RSA-W

3.18 Chapter Summary

The concepts and mathematical preliminaries used in RSA, ECC, RM-RSA and RA-ECC are discussed elaborately in this chapter. From the experimental results, it is observed that ECC takes more time than RSA for both operational time and power consumption. This is because lot of computations like generation of points for EC, addition of points in performing k[P] are involved in ECC. Normally, regular RSA and ECC takes more time. To reduce the operational time, the conventional RMs and RAs are used in RSA and ECC respectively. To reduce the operational time further, in both cryptographic algorithms which are used in mobile devices, ACs are incorporated. To generate the AC for the given integer, the BIAs viz., PSO, SSO and

BFO are taken in this thesis. If the energy required for the same gets reduced, the operational time too will come down which ultimately increases the life-time of the battery. They are discussed in the upcoming chapters.

CHAPTER - IV

GENERATION OF ADDITION CHAIN USING PARTICLE SWARM OPTIMIZATION

4.1 Background

On mobile or handheld computers, security is the key concern since the internet community can do its job anywhere at any time. To encrypt information on mobile devices, various cryptographic algorithms such as RSA, ECC etc., can be used today. But, they take some battery power, OS, memory size, processing speed, screen size, resolution, etc. Providing security for mobile devices with limited power and increasing the operational speed are difficult job. In order to have the security, cryptographic algorithms are being used. To minimize the operational time, the computation involved in encryption and decryption operations should be eased. Even though, many methods exist in literature to reduce the said time, ACs are more predominant one. This is because, it reduces the number of multiplications in RSA and the number of additions in ECC.

There are several methods exist in literature to generate the ACs for the given integer, the bio-inspired or Evolutionary Algorithms (EA) are taken in this work. The term computer intelligence or computational intelligence frequently used to refer to EAs. EAs are inspired by the idea of either natural evolution or social behaviour of insects, birds, animals etc. In this chapter, ACs are generated using PSO termed as AC-PSO and they are used in RSA and ECC with two different emulators for performing encryption and decryption operations where the said cryptosystems are used in mobile devices. They are used in encryption and decryption phases of RSA and ECC. Also,

the time taken for the said phases and the power consumption for the same are also evaluated.

4.2 Need for PSO Algorithm

PSO algorithm is a computational method that optimizes the solution by iteratively trying to improve the candidate solution. In PSO, each individual (particle) x_i is moving with some velocity through the search space which is the essence of PSO. As a PSO, individual moves through the search space, it has some inertia and so it tends to maintain its velocity. However, its velocity can change due to a couple of different factors viz., (i) it remembers its best position in the past, and it would like to change its velocity to return that position. Also, in PSO, an individual travels through the search space and its position in the search spaces changes from one generation to the next. However, the individual remembers its performance from past generations, and it remembers the search space location at which it is obtained its best performance in the past (ii) An individual knows the best position of its neighbours at the current generation. It requires the definition of neighbourhood size and it requires that all of the neighbours communicate with each other about their performance of the optimization problem [30]. It has no evolution operators.

It is noted that in AC, the first two numbers in AC is always 1 and 2, i.e., 1-2. From 2, there are 2 numbers viz., 3 and 4. Since there are two numbers, the search space also consists of two numbers. On the other hand, if the current number is 10, the search space consists of 10 numbers starting from 1 to 10 where the next number after 10 is obtained either using addition step or doubling step depending on the velocity. Since the velocity determines the neighbour or next number, the optimal AC is determined based on it. Since, finding optimal AC is an NP-hard, it is possible only using PSO.

An AC is said to be an optimal, its length should be minimum and also the time taken for encryption and decryption and the power consumption for the above said operations are less.

4.3 Concepts Used in PSO

In computational science, PSO is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It is a population-based optimization technique inspired by the motion of bird flocks and schooling fish. In PSO, all the birds do not know where food is but they know how they in each iteration. In PSO, each member of the population is called particle and the population is called swarm. PSO shares many similarities with evolutionary computation techniques. The system is initialized with a population of random solutions, and the search for the optimal solution is performed by updating generations. PSO has no evolution operators, such as crossover and mutation. In PSO, the potential solutions, called particles, move in the problem space by following the current optimum particles. It is computationally more efficient in terms of both speed and memory requirements.

Dr. Eberhart and Dr. Kennedy [32][33] proposed PSO in 1995 on the basis of flocking birds' social activity and fish schooling. It is a meta-heuristic algorithm. With a population of multiple random solutions, it is initialised. The effects are refined for the best outcome by iterations and by tiding generations it acquires an optimal solution. Although, a group of birds in an area are looking for food, their initial locations are random. The birds initially do not know the location of the food. But after a set of movements, which are iterations, they get closer to the food. Following the direction of the bird that is closest to food is the fastest way to reach food. All particles are modified on the basis of the two best values after each iteration. The first

best value called $pBest$ is already obtained by a particle. The second-best value called g_{best} is the best value achieved for the fitness function tracked by the particle swarm optimizer by the general population.

When the entire group is searching for a certain target, for one individual, the individual in the current optimal position of the group and the optimal position that it has reached is often referenced to adjust the next search. Eberhart and Kennedy modified the model of this simulated group interaction and designed it as a general method to solve optimization problems; they called it as PSO algorithm. There is no parameter controlling the progression of p_{best} values in general flow of PSO. In other words, there exists the need of a parameter regenerating the insufficient particles that cannot improve their individual best position value (p_{best}). On the other hand, velocity and position concepts perform the update of particles as much as employed. Since each parameter tries to modify the position by using the information viz., (i) the current position, (ii) the current velocity (iii) the distance between the current position and pbest (iv) the distance between the current position and g_{best}. The new velocity and new positions are calculated using eqn. (4.1) and eqn. (4.2) respectively.

$$\forall i \in \{1, n\}: V_i = V_i + c_1 \times \gamma_1 \times (Pb_i - P_i) + c_2 \times \gamma_2 \times (Gb_i - P_i) \quad \text{... (4.1)}$$

where

n : number of maximum permitted iterations

V : velocity of the particle

Pb: *pbest* (Particle best)

P: present position

Gb: *gbest* (Global best)

c_{1i}, c_2 : learning factors or accelerating factors related to p_{best}, and g_{best} respectively. In general $c_1 = c_2$ selected from the range of 0 to 1.

γ_1,γ_2: random numbers between 0 to 1.

$$P = P + V \quad \text{... (4.2)}$$

PSO emulates the interaction between members to share information. It has been applied to numerous areas in optimization and in combination with other existing algorithms. This method performs the search of the optimal solution through agents, referred to as particles, whose trajectories are adjusted by a stochastic and a deterministic component. Each particle is influenced by its 'best' achieved position and the group 'best' position, but tends to move randomly. A particle i is defined by its position vector, x_i, and its velocity vector, V_i. Every iteration, each particle changes its position according to the new velocity as in eqn.(4.1) where p_{best} and g_{best} denote the best particle position and best group position and the parameters c_1, c_2, r_1 and r_2 are respectively inertia weight, two positive constants and two random parameters within $[0, 1]$. Usually maximum and minimum velocity values are also defined and initially the particles are distributed randomly to encourage the search in all possible locations.

One of the advantages of PSO over other derivative-free methods is the reduced number of parameters to tune and constraints acceptance. A 2D representation of one particle, 'i', movement between two positions. It can be observed how the particle best position, p_{best}, and the group best position, g_{best}, influence the velocity of the particle at the next iteration. Nevertheless, the stochastic properties of the algorithm allow for solution variability to guarantee the solution space exploitation. Fig. 4.1 shows the movement of the particle 'i' in the solution space during

iterations k and $k + 1$. The evolution of the particle movement is influenced by the particle best position, p_{best}, and the group best position, g_{best}.

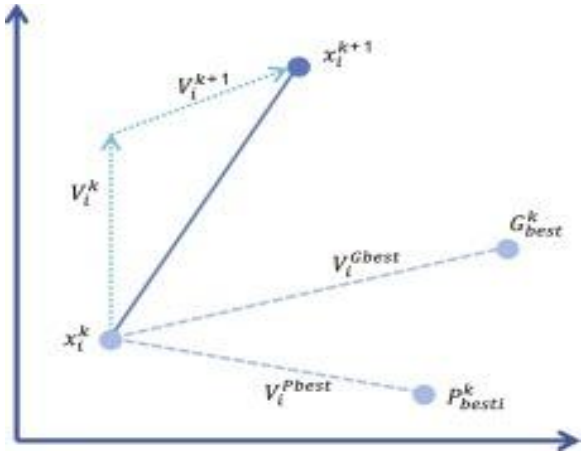


Fig. 4.1: Movement of the particle *'i'* in the solution space during iterations k and $k + 1$.

In this work, particle represents the AC. Velocity V_i represents the number to be added to the current number x_i so that the next number x_{i+1} is obtained. V_i is obtained eqn. (4.2). It corresponds to either by using addition step or doubling step used in AC depending on the random number chosen. Further, fitness function is taken as length of AC denoted as $l(x_{i+1})$. In this work, $c_1=c_2=0.7$ where 0.7 is a uniform random number. Similarly, other random numbers r_1, r_2 are taken from RAND corporation table. When RSA and ECC are considered, the key is taken very large, and AC of the key is generated according to the proposed AC- PSO. The steps involved in PSO are shown in Pseudo code 4.1.

Pseudo Code 4.1: PSO

---

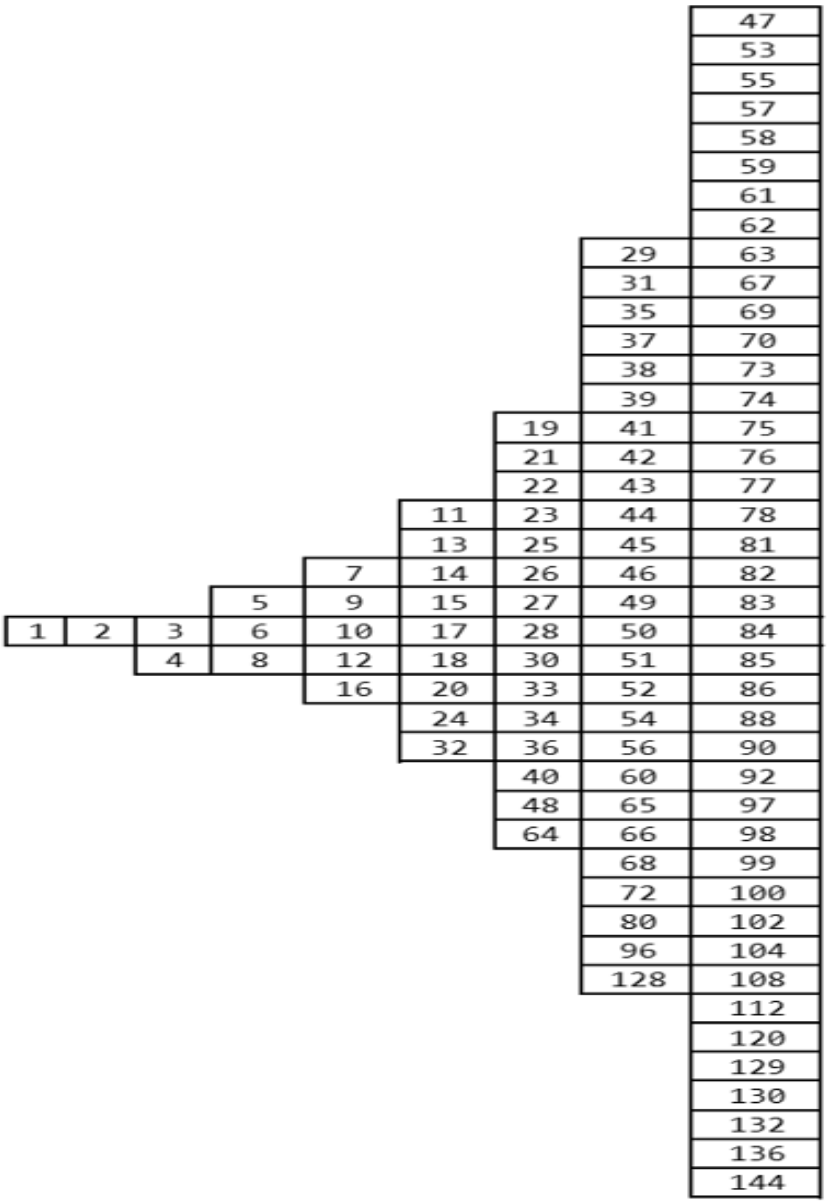
a. Initialize all particles
b. Calculate fitness value for each particle
c. If the calculated fitness value p_{best} is better than existing p_{best}, then update p_{best}
d. Find the particle with best fitness value g_{best} from overall population

e. For all particles, calculate velocity and position based on the equations and update values
f. Repeat from Step b until maximum number of iterations achieved or optimum result achieved.

4.4 Proposed AC-PSO Methodology

In order to generate the AC for the given integer n, it is noted that the first number is always 1. Let it be x_1. The next number in AC is 2. Let it be x_2. This is because 2 is obtained from 1 either by using addition or doubling step of x_1. Let it be x_2. From 2, the number 3 $(2+1=3)$ is obtained by using addition step and 4 is obtained either addition step $(3+1)$ or doubling step $2(2)=4)$. Thus $x_3 \in \{3,4\}$ and the corresponding AC is $1-2-3$ or $1-2-4$ and its $l(x_3) = 2$. From 3, the numbers $4(3+1), 5(3+2)\ and\ 6(3+3)$ are obtained and the ACs are (i) $1-2-3-4$ (ii) $1-2-3-5$ (iii) $1-2-3-5$ with length 4. But, the AC is not considered as optimal length AC because the AC for 4 has been generated previously with length 3. Similarly, from 4, the next numbers $5\ (1-2-4-5)$, $6\ (1-2-4-6)$ and $8\ (1,2,4,8)$. Let it be x_4. Thus, $x_4 \in \{5,6,8\}$.

To generate, the next number from $x_4 = 5$, they are $6(5+1), 7(5+2), 8(5+3)$ with their lengths 4, i.e., the ACs are $1-2-3-5-6, 1-2-3-5-7, 1-2-3-5-8, 1-2-3-5-10$. But, the AC 1 – 2 – 3 – 5 - 6 is not considered due to increasing its length. The numbers in x_i, $i = 5, \ldots, n-1$ are generated in this manner and they are shown fig. 4.1. From fig. 4.1, it is observed that in general if the number is say i, if $i \in x_n$, then $l(i) = n-1$. For example, if $i = 76$, then $l(76) = 9-1 = 8$. This concept is very useful in generating the AC for any number n. Fig. 4.2 shows the numbers occur in P_i^{th} position.



x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8

Fig. 4.2: Numbers Occur in P_i , i = 1, 2, ...8 Without Duplication

4.5 Generation of AC-PSO - An Example

In order to generate the AC using PSO, let $w = 0.9$, $c_1 = c_2 = 1.5$ and r_1, r_2 are taken from RAND table, and fitness function is $l(x_i)$, $i = 0,1,2,3, \ldots$. For example, to generate AC for the integer 20 i.e., $n = 20$. Since $x_1 = 1$; $x_2 = 2$. To generate the numbers for x_3 , Now v_k, $k = 3,4$ are computed using eqn. (4.1) as

$$v_{33} = 0.9(0.1116) + 1.5\,(0.4363)\,(3-2) + 1.5\,(0.1875)\,(3-2) = 1.0361$$
$$= \lfloor 1.0361 \rfloor = 1$$

$x_4 = 3 + 1 = 4 \in x_4$. Thus, the AC for 5 is $1 - 2 - 3 - 5$ and fitness value $l(5) = 3$.

$$v_{34} = 0.9(0.0613) + 1.5(0.7674)(4-2) + 1.5(0.2632)(4-2) = 3.14697$$
$$= \lfloor 3.14697 \rfloor = 3$$

$x_4 = 4 + 3 = 7 \in x_4$.

Thus, the AC for 7 is $1 - 2 - 4 - 7$ and fitness value $l(7) = 3$.

Table 4.1 shows the generation of AC for the integer $n = 10$ based on PSO. Only for illustration purpose small integer is taken. But, when RSA and ECC are considered, the key is taken very large, and AC of the key is generated according to the proposed PSO-AC.

Table 4.1: Generation of AC for n=10 Using AC-PSO

i	x_i	$AC(x_i)$	$l[AC(x_i)]$	Random Nos Taken for			Computation of V_i using eqn. (4.1)	$IV_i = \{\lfloor V_i \rfloor, \lceil V_i \rceil\}$
				rv_i	r_1	r_2		
3	3	1-2-3	2	0.1116	0.4363	0.1875	1.0361	{1,2}
	4	1-2-4	2	0.1116	0.4363	0.1875	1.9719	{1,2}
4	5	1-2-3-5	3	0.0613	0.7674	0.2632	3.1290	{3,4}
	5	1-2-4-5	3	0.0613	0.7674	0.2632	1.6018	{1,2}
	6	1-2-4-6	3	0.0613	0.7674	0.2632	3.1287	{3,4}
5	7	1-2-3-5-7	4	0.0751	0.0010	0.4312	1.3642	{1,2}
	10	1-2-4-6-10	4	0.0751	0.0010	0.4312	2.6608	{2,3}
5	7	1-2-3-5-7	4	0.0418	0.1922	0.8917	3.2893	{3,4}
	10	1-2-4-6-10	4	0.0418	0.1922	0.8917	6.5412	{6,7}

-
-
-

$x_{i+1} = x_i + v_i$	Is $x_{i+1} \in p_i\,[x_{i+1}]$	$AC(x_{i+1})$	Is valid $AC(x_{i+1})$	$l[AC(x_{i+1})]$	x_{old}^{i+1}	ϕ_{best}^{i+1}	g_{best}^{i+1}
{4,5}	4∉{5,6,8} 5∈{5,6,8}	- 1-2-3-5	- Y	- 3	- 3	- 5	- 5
{5,6}	5∈{5,6,8} 6∈{5,6,8}	1-2-4-5 1-2-4-6	Y Y	3 3	4 4	5 6	5 6
{8,9}	8∉{7,9,10,12,16} 9∈{7,9,10,12,16}	- 1-2-3-5-9	- N	- -	- -	- -	- -
{6,7}	6∉{7,9,10,12,16} 7∈{7,9,10,12,16}	- 1-2-4-5-7	- Y	- 4	- 5	- 7	- 7
{9,10}	9∉{7,9,10,12,16} 10∈{7,9,10,12,16}	1-2-4-6-9 1-2-4-6-10	N Y	- 4	- 6	- 10	- 10
{8,10}	8∉{11,13,14,15,16,17}1 0∉{11,13,14,15,17,19}	- -	- -	- -	- -	- -	- -
{12,13}	12∉{11,13,14,15,17,18} 13∈{11,13,14,15,16,18}	- 1-2-4-6-10-13	- N	- -	- -	- -	- -

4.6 Proposed AC-PSO Based Cryptosystem

In order to speed up the operational time, reducing the power consumption and higher security, ACs for the given integer are generated based on PSO and they are incorporated into RSA and ECC.

4.6.1 AC-PSO-RSA and AC-PSO-ECC Methodology

The main operations of RSA are encryption/decryption which consists of modular exponentiation (ME). They involve raising to the powers. This process involves many multiplications (M) which make it time consuming. For example, to compute x^e based on the process of adding and multiplying, it needs $(e-1)$ RMs of x. Similarly, in ECC, scalar point multiplication $k[P]$ mod m, where P is a point on EC, k is an arbitrary integer, and m is a modulus which plays a crucial role. To perform $k[P]$, i.e., $(k-1)$ RAs of P are needed.

- **Proposed AC-PSO-RSA Methodology - An Example**

In order to fully understand the topic under study using RSA, discussed in section 3.4.1 of Chapter III. Let $p = 13, q = 17$ and $e = 11$, then $n = 13(17) =$

$221, (p-1)(q-1) = 12(16) = 192$. Then using extended Euclidean algorithm d is computed as $d = 133$. To encrypt, i.e., $C = M^{11} \bmod 187$, which requires 10 RMs. However, if the AC-PSO is used, the AC for $e = 11$ is $1-2-3-5-10-11$; $l(e) = 5$ which needs, only 5 Ms.

- **Proposed AC-PSO-ECC Methodology - An Example**

Consider the EC, $y^2 \equiv x^3 + 17x + 7 \; mod \; 539039$. Using Diffie-Hellman key exchange protocol as discussed in section 3.10 of Chapter III. A's private key $n_A = 65131$ and the base point $p_B = (2,7)$. Thus A's public key $P_A = n_A P_B = 65131(2,7)$. When P_A is performed by RAs, it requires 65130 additions. To reduce the number of additions, PSO based AC is used and one of the ACs for 65131 is $1-2-3-5-10-20-40-43-83-166-249-498-996-1992-2035-4070-8140-16280-32560-32565-65130-65131$, i.e., $l(65131) = 21$. Thus, it requires only 21 additions. Based on this AC, the addition of EC points are performed using section 3.7.2 of Chapter III. Let $P = (2, 7)$. Then,

$2P = P + P$ $= (244768, 340039)$	$3P = 2P + P$ $= (11724, 249063)$	$5P = 3P + 2P$ $= (506411, 102155)$
$10P = 2(5P)$ $= (219797, 239709)$	$20P = 2(10P)$ $= (380891, 525867)$	$40P = 2(20P)$ $= (214059, 41583)$
$43P = 40P + 3\,P$ $= (8793, 297511)$	$83P = 43P + 40\,P$ $= (137967, 526592)$	$166P = 83P + 83P$ $= (382077, 526592)$
$249P = 166P + 83P$ $= (217586, 32021)$	$498P = 2(249P)$ $= (221645, 21083)$	$996P = 2(498P)$ $= (201219, 179442)$
$1992P = 2(996P)$ $= (84103, 159148)$	$2035P = 1992P + 43P$ $= (415947, 244729)$	$4070P = 2(2035P)$ $= (48695, 28739)$
$8140P = 2(4070P)$ $= (11378, 429659)$	$16280P = 2(8140P)$ $= (3661662, 126957)$	$32560P = 2(32560P)$ $= (6996, 35500)$
$32565P = 32560P + 5P$ $= (314717, 375063)$	$65130P = 2(32560P)$ $= (104163, 454622)$	$65131P = 65130P + P$ $= (350818, 39982)$

4.7 Results and Discussion

The proposed methodology is implemented in VC++ with Android and Windows emulator for varying file sizes using RSA and ECC. The ET, DT (in mS), EP, DP (in mW) are computed and SE (in %) is measured by ABC Hackman tool. The results obtained from the implementation are recorded from table 4.2 to 4.11 and their corresponding graphical representations are shown from fig. 4.3 to 4.12.

Table 4.2: Encryption Time (mS) using AC-PSO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-PSO based RSA and ECC	
	without RM and RA		with RM and RA			
	ET-RSA-A	ET-ECC-A	ET-RM-RSA-A	ET-RA-ECC-A	ET-AC-PSO-RSA-A	ET-AC- PSO-ECC-A
1	1660	2447	1227	1820	1041	1578
2	3237	4790	2377	3555	2048	3069
4	6494	9553	4788	7173	4099	6150
8	13689	20179	10077	15130	8644	12954
16	27426	40415	20199	30326	17322	25981
Total	52506	77384	38668	58004	33154	49732
Avg.	10501.2	15476.8	7733.6	11600.8	6630.8	9946.4

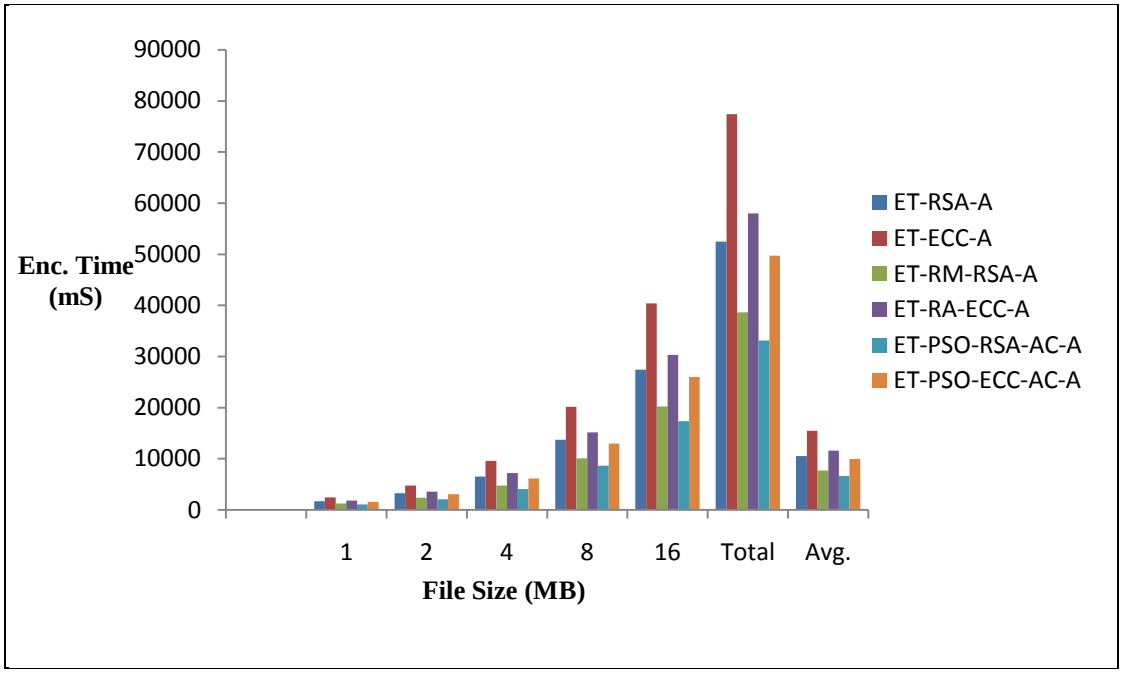


Fig. 4.3: Graph showing the Encryption Time (mS) using AC-PSO in RSA and ECC with Android Emulator

- ET-AC-PSO-RSA-A is 1.584 times faster than ET-RSA-A and 1.166 times faster than ET-RM-RSA-A
- ET-AC-PSO-ECC-A is 1.556 times faster than ET-ECC-A and 1.166 times faster than ET-RA-ECC-A
- ET-AC-PSO-RSA-A is 1.500 times faster than ET-AC- PSO-ECC-A

Table 4.3: Decryption Time(mS) using AC-PSO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-PSO based RSA and ECC	
	without RM and RA		with RM and RA			
	DT-RSA-A	DT-ECC-A	DT-RM - RSA-A	DT-RA-ECC-A	DT-AC-PSO-RSA-A	DT-AC- PSO-ECC-A
1	1616	2330	1205	1789	1041	1544
2	3193	4519	2352	3519	2011	3040
4	6490	9209	4776	7171	4102	6156
8	13645	19375	10066	15082	8623	12925
16	27399	38936	20193	30294	17301	25954
Total	52343	74369	38592	57855	33078	49619
Avg.	10468.6	14873.8	7718.4	11571	6615.6	9923.8

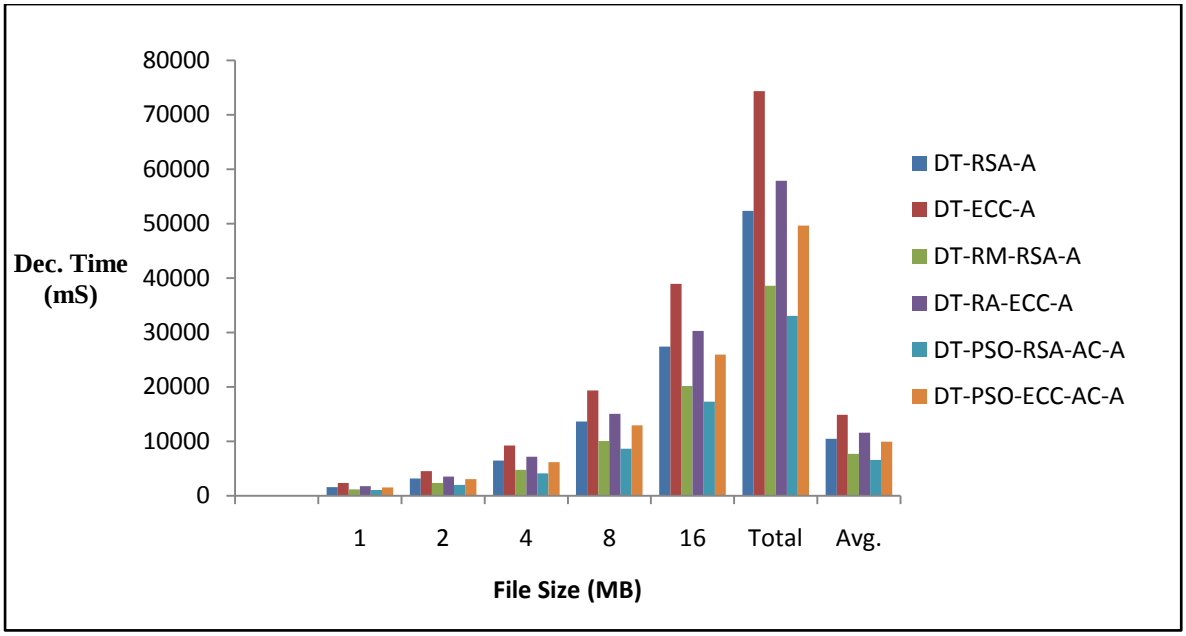


Fig. 4.4: Graph showing Decryption Time (mS) using AC-PSO in RSA and ECC with Android Emulator

- DT-AC-PSO-RSA-A is 1.582 times faster than DT -RSA-A and 1.167 times faster than DT -RM-RSA-A
- DT -AC-PSO-ECC-A is 1.498 times faster than DT -ECC-A and 1.165 times faster than DT -RA-ECC-A
- DT -AC-PSO-RSA-A is 1.500 times faster than DT -AC- PSO-ECC-A

Table 4.4: Encryption Power (mW) using AC-PSO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-PSO based RSA and ECC	
	without RM and RA		with RM and RA			
	EP-RSA-A	EP-ECC-A	EP-RM-RSA-A	EP-RA-ECC-A	EP-AC-PSO-RSA-A	EP-AC-PSO-ECC-A
1	554	817	421	613	357	552
2	1102	1621	806	1197	682	1042
4	2171	3208	1612	2410	1372	2073
8	4569	6731	3364	5042	2893	4325
16	9156	13510	6735	10134	5788	8670
Total	17552	25887	12938	19396	11092	16662
Avg.	3510.4	5177.4	2587.6	3879.2	2218.4	3332.4

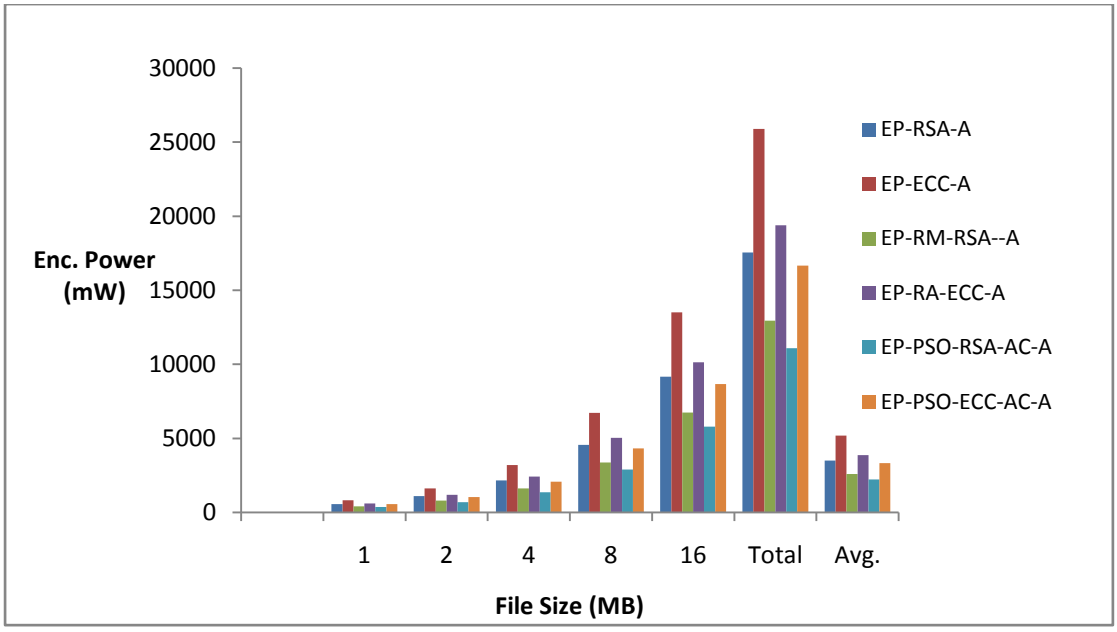


Fig. 4.5: Graph showing the Encryption Power (mW) using AC-PSO in RSA and ECC with Android Emulator

- EP-AC-PSO-RSA-A is 1.582 times less than EP -RSA-A and 1.166 times less than EP -RM-RSA-A.
- EP -AC-PSO-ECC-A is 1.553 times less than EP -ECC-A and 1.164 times less than EP -RA-ECC-A
- EP -AC-PSO-RSA-A is 1.502 times less than EP -AC- PSO-ECC-A

Table 4.5: Decryption Power (mW) using AC-PSO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-PSO based RSA and ECC	
	without RM and RA		with RM and RA			
	DP-RSA-A	DP-ECC-A	DP-RM-RSA-A	DP-RA-ECC-A	DP-AC-PSO-RSA-A	DP-AC-PSO-ECC-A
1	562	815	411	602	354	534
2	1081	1506	802	1178	673	1038
4	2175	3072	1608	2417	1375	2066
8	4548	6463	3357	5035	2874	4330
16	9148	12995	6743	10109	5772	8674
Total	17514	24851	12921	19341	11048	16642
Avg.	3502.8	4970.2	2584.2	3868.2	2209.6	3328.4

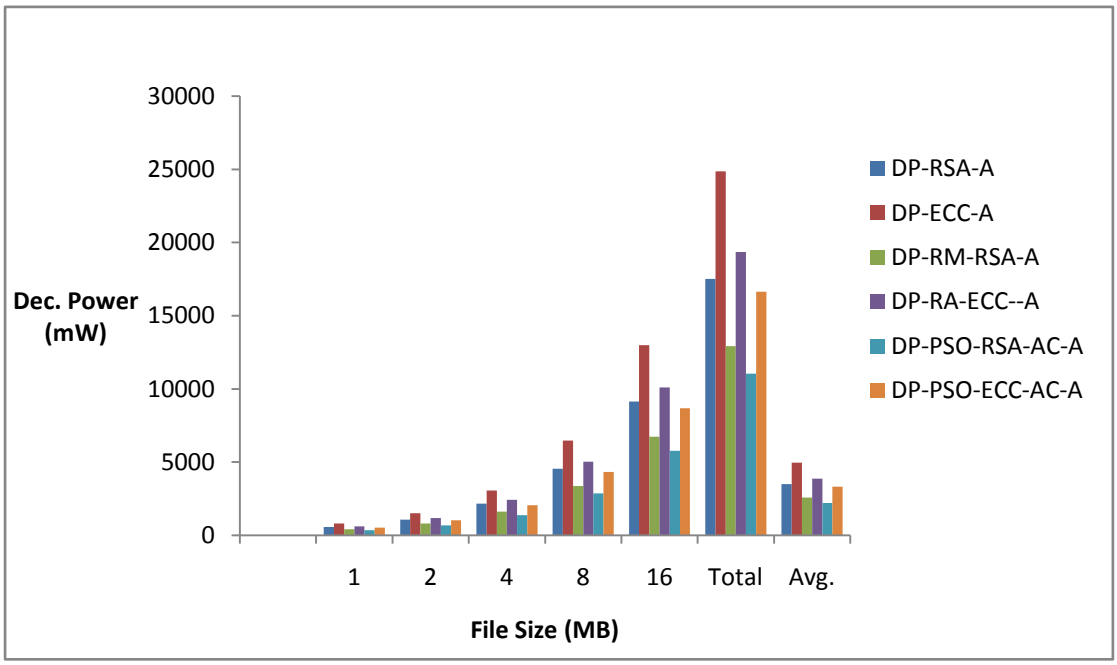


Fig. 4.6: Graph showing the Decryption Power (mW) using AC-PSO in RSA and ECC with Android Emulator

- DP-AC-PSO-RSA-A is 1.585 times less than DP -RSA-A and 1.169 times less than EP -RM-RSA-A.
- DP -AC-PSO-ECC-A is 1.493 times less than DP -ECC-A and 1.162 times less than DP -RA-ECC-A
- DP -AC-PSO-RSA-A is 1.506 times less than DP -AC- PSO-ECC-A

Table 4.6: Security (%) using AC-PSO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-PSO based RSA and ECC	
	without RM and RA		with RM and RA			
	SE-RSA-A	SE-ECC-A	SE-RM-RSA-A	SE-RA-ECC-A	SE-AC-PSO-RSA-A	SE-AC-PSO-ECC-A
1	89	93	92	94	94	94
2	88	89	89	92	91	93
4	87	88	88	90	90	91
8	85	88	87	90	89	90
16	85	86	87	89	88	90
Avg.	86.8	88.8	88.6	91	90.4	91.6

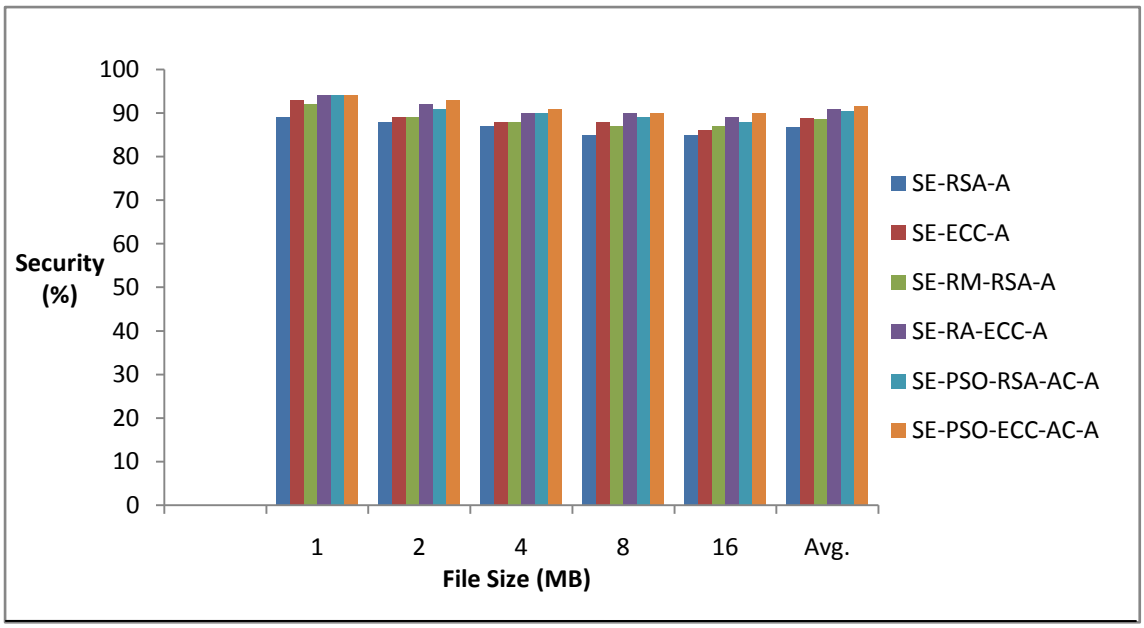


Fig. 4.7: Graph showing the Security (%) of AC-PSO in RSA and ECC with Android Emulator

- SE-AC-PSO-RSA-A is 1.041 times more than SE-RSA-A and 1.020 times more than SE-RM-RSA-A
- SE-AC- PSO-ECC-A is 1.032 times more than SE-ECC-A and 1.007 times more than SE-RA-ECC-A
- SE-AC- PSO-ECC-A is 1.013 times more than SE-AC-PSO-RSA-A

Table 4.7: Encryption Time (mS) using AC-PSO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-PSO based RSA and ECC	
	without RM and RA		with RM and RA			
	ET-RSA-W	ET-ECC-W	ET-AC-RSA-W	ET-AC-ECC-W	ET-AC-PSO-RSA-W	ET-AC- PSO-ECC-W
1	1654	2441	1204	1728	1046	1520
2	3233	4748	2378	3362	2050	2984
4	6490	9559	4775	6753	4096	6008
8	13670	20174	10082	14252	8652	12693
16	27432	40443	20214	28603	17318	25402
Total	52479	77365	38653	54698	33162	48607
Avg.	10495.8	15473	7730.6	10939.6	6632.4	9721.4

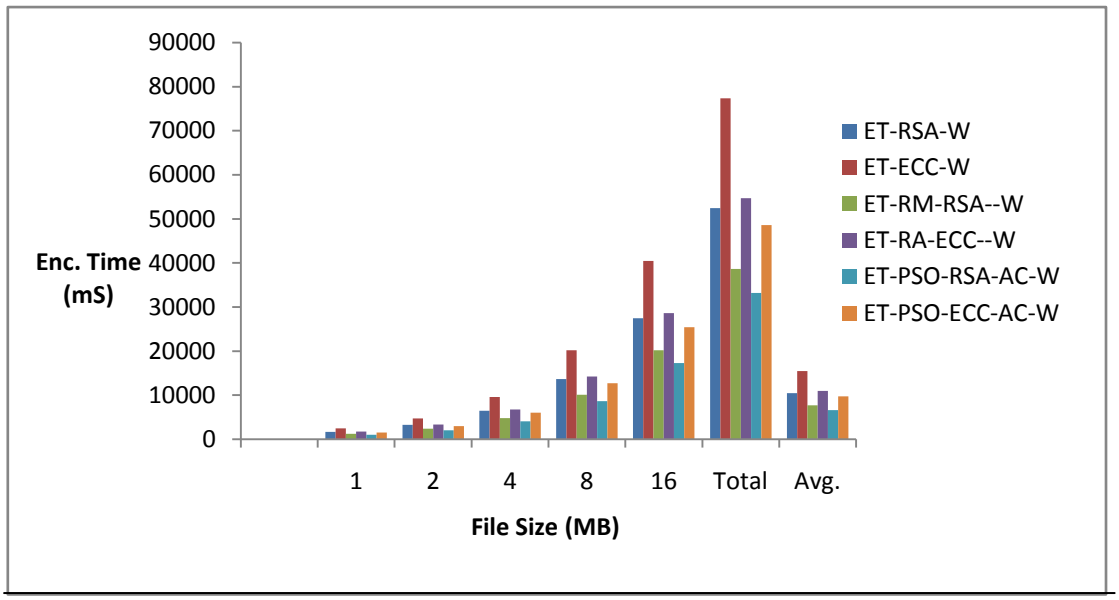


Fig. 4.8: Graph showing the Encryption Time(mS) using AC-PSO in RSA and ECC with Windows Emulator

- ET-AC-PSO-RSA-W is 1.582 times faster than ET-RSA-W and 1.165 times faster than ET-RM-RSA-W
- ET-AC-PSO-ECC-W is 1.591 times faster than ET-ECC-W and 1.125 times faster than ET-RA-ECC-W
- ET-AC-PSO-RSA-W is 1.465 times faster than ET-AC- PSO-ECC-W

Table 4.8: Decryption Time (mS) using AC-PSO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-PSO based RSA and ECC	
	without RM and RA		with RM and RA			
	DT-RSA-W	DT-ECC-W	DT-AC-RSA-W	DT-AC-ECC-W	DT-AC-PSO-RSA-W	DT-AC-PSO-ECC-W
1	1616	2322	1205	1708	1027	1517
2	3184	4530	2366	3336	2026	2967
4	6496	9212	4778	6755	4099	6017
8	13657	19396	10050	14240	8616	12647
16	27401	38958	20210	28571	17310	25388
Total	52354	74418	38609	54610	33078	48536
Avg.	10470.8	14883.6	7721.8	10922	6615.6	9707.2

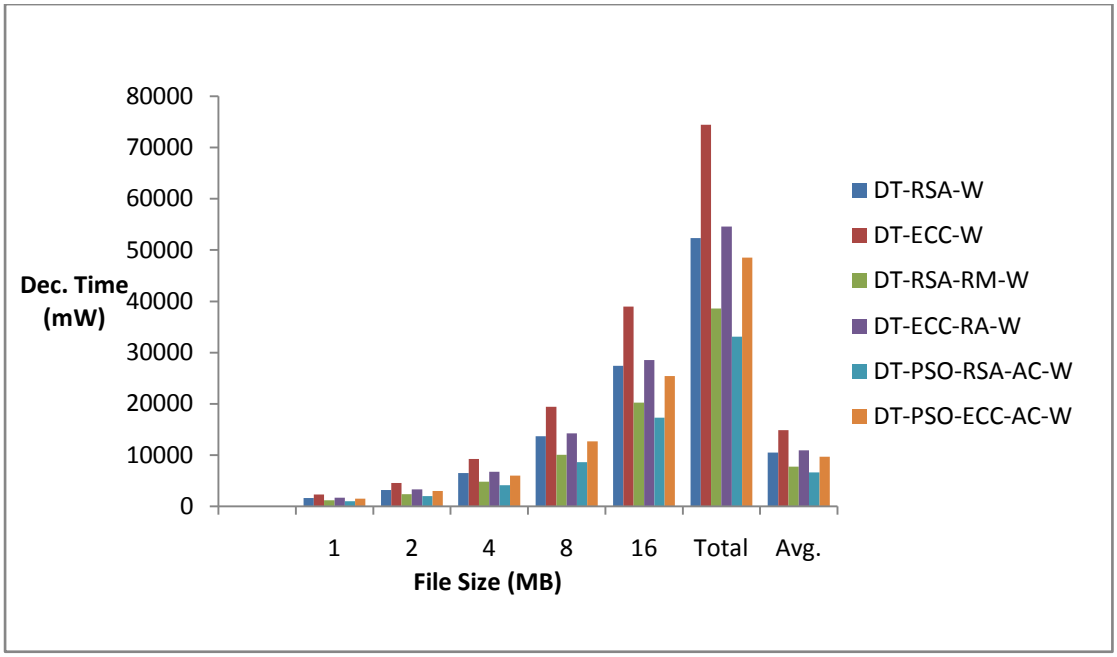


Fig. 4.9: Graph showing the Decryption Time (mS) using AC-PSO in RSA and ECC with Windows Emulator

- DT-AC-PSO-RSA-W is 1.582 times faster than DT -RSA-W and 1.167 times faster than DT -RM-RSA-W
- DT -AC-PSO-ECC-W is 1.533 times faster than DT -ECC-W and 1.125 times faster than DT -RA-ECC-W
- DT -AC-PSO-RSA-W is 1.467 times faster than DT -AC- PSO-ECC-W

Table 4.9: Encryption Power (mW) using AC-PSO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-PSO based RSA and ECC	
	without RM and RA		with RM and RA			
	EP-RSA-W	EP-ECC-W	EP-AC-RSA-W	EP-AC-ECC-W	EP-AC-PSO-RSA-W	EP-AC-PSO-ECC-W
1	571	840	421	592	348	513
2	1100	1582	796	1146	699	1003
4	2179	3228	1604	2263	1381	2029
8	4577	6750	3361	4765	2893	4246
16	9165	13496	6738	9559	5775	8483
Total	17592	25896	12920	18325	11096	16274
Avg.	3518.4	5179.2	2584	3665	2219.2	3254.8

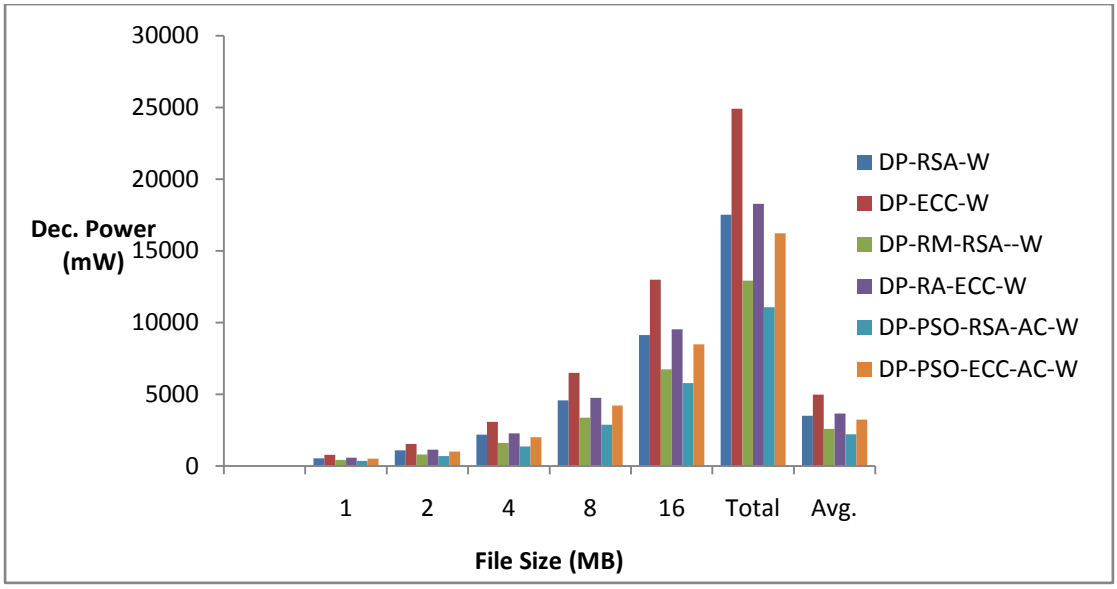


Fig.4.10: Graph showing the Encryption Power (mW) using AC-PSO in RSA and ECC with Windows Emulator

- EP-AC-PSO-RSA-W is 1.585 times less than EP -RSA-W and 1.164 times less than EP -RM-RSA-W
- EP -AC-PSO-ECC-W is 1.591 times less than EP -ECC-W and 1.126 times less than EP -RA-ECC-W
- EP -AC-PSO-RSA-W is 1.467 times less than EP -AC- PSO-ECC-W

Table 4.10: Decryption Power (mW) using AC-PSO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-PSO based RSA and ECC	
	without RM and RA		with RM and RA			
	DP-RSA-W	DP-ECC-W	DP-AC-RSA-W	DP-AC-ECC-W	DP-AC-PSO-RSA-W	DP-AC-PSO-ECC-W
1	541	785	415	584	350	510
2	1086	1536	805	1140	693	1010
4	2192	3088	1604	2275	1368	2008
8	4579	6496	3368	4761	2877	4220
16	9133	12997	6742	9529	5778	8472
Total	17531	24902	12934	18289	11066	16220
Avg.	3506.2	4980.4	2586.8	3657.8	2213.2	3244

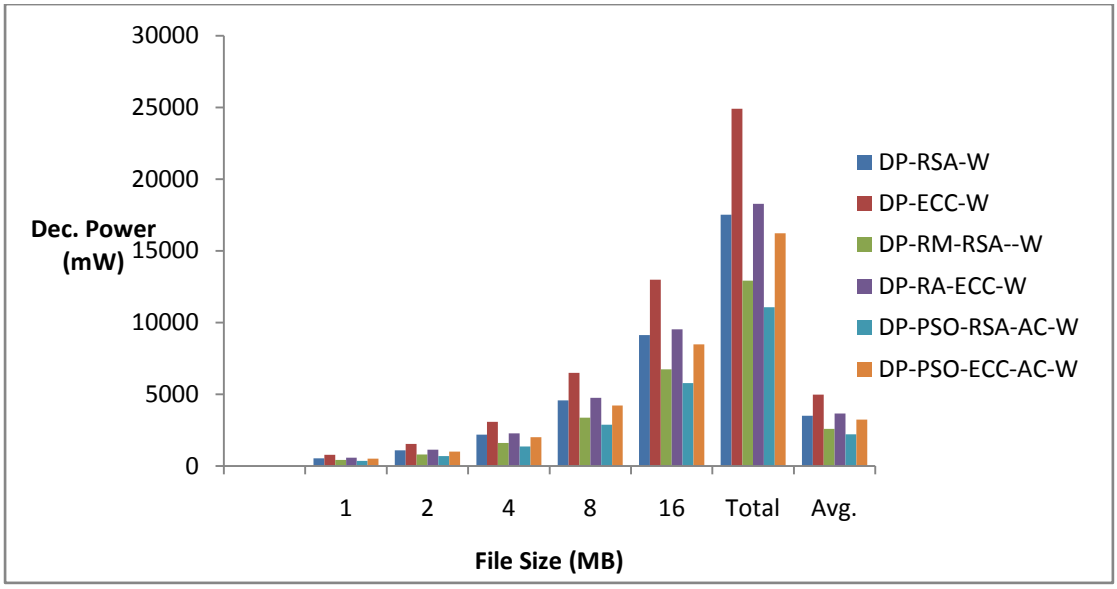


Fig.4.11 : Graph showing the Decryption Power (mW) using AC-PSO in RSA and ECC with Windows Emulator

- DP-AC-PSO-RSA-W is 1.584 times less than DP -RSA-W and 1.168 times less than DP -RM-RSA-W
- DP -AC-PSO-ECC-W is 1.535 times less than DP -ECC-W and 1.127 times less than DP -RA-ECC-W
- DP -AC-PSO-RSA-W is 1.466 times less than DP -AC- PSO-ECC-W

Table 4.11: Security (%) using AC-PSO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-PSO based RSA and ECC	
	without RM and RA		with RM and RA			
	SE-RSA-W	SE-ECC-W	SE-AC-RSA-W	SE-AC-ECC-W	SE-AC-PSO-RSA-W	SE-AC-PSO-ECC-W
1	91	92	92	94	92	96
2	88	89	89	92	91	92
4	86	89	88	91	90	91
8	86	88	88	89	89	91
16	85	87	87	89	87	90
Avg.	87.2	89	88.8	91	89.8	92

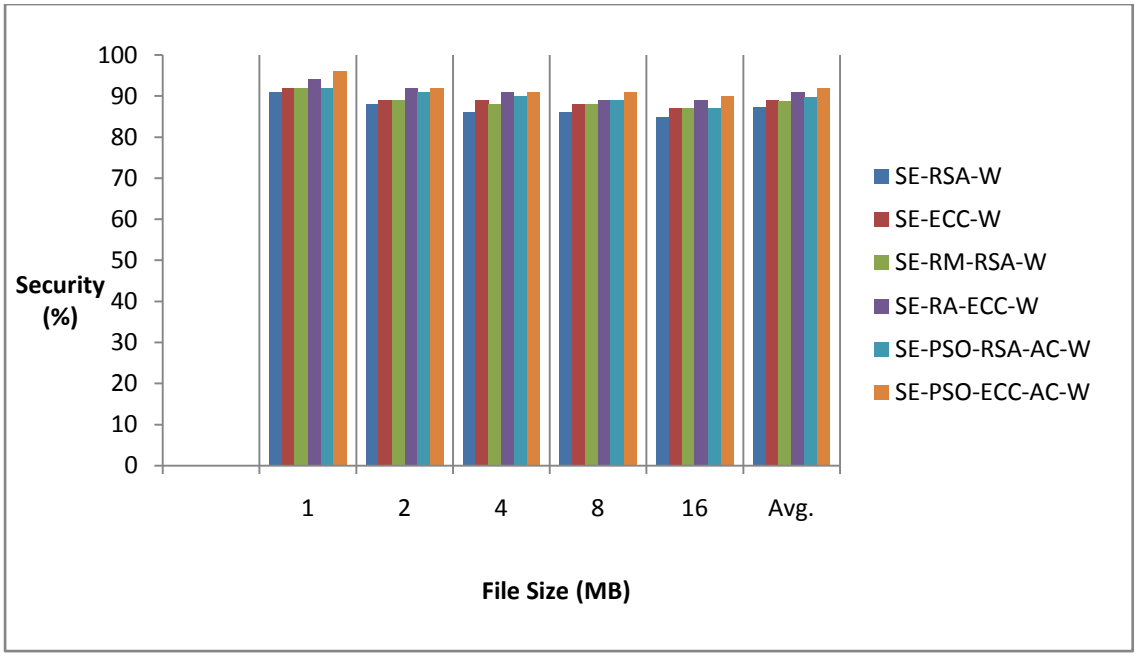


Fig.4.12: Graph showing the Security (%) using AC-PSO in RSA and ECC with Windows Emulator

- SE-AC-PSO-RSA-W is 1.030 times more than SE-RSA-W and 1.011 times more than SE-RM-RSA-W
- SE-AC- PSO-ECC-W is 1.034 times more than SE-ECC-W and 1.011 times more than SE-RA-ECC-W
- SE-AC- PSO-ECC-W is 1.024 times more than SE-AC-PSO-RSA-W

From the above results, ECC takes more time than RSA for both operational and power consumption time. There is because ECC is computationally more intensive approach than RSA. The operational time and power consumption when using AC-based PSO is substantially reduced. This shows the efficiency of AC based PSO.

4.8 Chapter Summary

ACs based on PSO are taken into account, integrated into RSA and ECC. They are implemented successfully. From the experimental results, it is observed that AC-PSO-RSA takes less operational time, consumes less power than AC-RSA and AC-RM-RSA when using both emulators. Higher protection levels are achieved by AC-PSO-ECC when considering security in AC-ECC and AC-RA-ECC. It is also advised to

use AC-PSO-RSA when there is a small power source for a mobile device to run. AC-PSO-ECC offers security of 92% approximately when security parameter is considered. It is concluded that the experimental findings have clearly shown that the proposed AC-PSO with RSA and ECC cryptography systems can be used either to decrease operating power or to achieve enhanced security that are the primary motive of this work. It is noted that the time taken for all the parameters are decreasing when PSO is used due to the velocity and position play a vital role to generate the AC. To reduce the time further for the said parameters without compromising the optimal length AC for the given integers, an another BIA SSO is considered and it is discussed in next chapter.

CHAPTER – V

GENERATION OF ADDITION CHAIN USING SIMPLIFIED SWARM OPTIMIZATION

5.1 Background

Mobile device applications have been enhanced its security by using cryptographic algorithms like RSA and ECC. Even though it has been improved using some EAs like PSO with AC in the previous chapter, a novel AI based algorithm namely Simplified Swarm Optimization (SSO) is taken in this chapter to minimize the time required for encryption and decryption process. SSO algorithm is also used to generate the optimal AC and it is termed as AC-SSO. In this chapter too, once the AC is generated using SSO, it is incorporated into RSA and ECC. The results are compared with existing RM-RSA and RA-ECC.

5.2 Need for AC-SSO

The major difference among SSO and other soft computing algorithms are their update mechanism (UM). For example, the UM of GA requires genetic operations like crossover and mutation. The UM of PSO needs to calculate both velocity and position via functions. But, they are not so in AC-SSO. Here, UM is based on only random number. Moreover, the UM of SSO is on NP - hard problem. There is no exact algorithm available to compute on exact solution to the NP - hard problem in polynomial time, soft computing has been widely used for that [77].

5.3 Concepts Used in SSO

SSO is a population-based, evolutionary, stochastic optimization technique in soft computing and it was originally designed by Yeh [31]. It has some advantages, such as fast convergence rate, few parameters, and easy implementation. It has simple

procedures and more powerful global searching, prevents from trapping local optimal procedures.

Let $X_i^t = (x_{i1}^t, x_{i2}^t, x_{i3}^t, \ldots, x_{ij}^t)$ be the i^{th} solution at the generation t, where c is the value of the j^{th} variable of

$$X_i^t p_i = (p_{i1}, p_{i2}, \ldots . p_{ij}) \quad \ldots (5.1)$$

represents the best solution with the best fitness value in its own history, known as $pbest$. The best solution with the best fitness value among all solutions is called $gbest$, which is denoted by

$$g = (g_1, p_2, \ldots . g_j) \quad \ldots (5.2)$$

and g_j denotes the j^{th} variable in $gbest$. x is a new randomly generated value between the lower bound and the upper bound is a uniform random number between [0, 1]. $C_w, C_p \; and \; C_g$ are three pre-defined parameters which form four interval probabilities representing the probabilities of the new variable updated from four sources, namely, the current solution, $pbest$, $gbest$ and a random movement in the problem space. The newly generated value is computed using eqn. (5.3).

$$x_{ij}^t = \begin{cases} x_{ij}^{t-1}, \; if \; rand() \in [0, \; C_w) \\ p_{ij}^{t-1}, \; if \; rand() \in \left[C_w, C_p\right) \\ g_j, \quad if \; rand() \in \left[C_p, C_g\right) \\ x, \quad if \; rand() \in \left[C_g, 1\right) \end{cases} \quad \ldots (5.3)$$

The primary steps of the SSO are shown in pseudo code 5.1

Pseudo Code: 5.1: SSO

---

1. Initialize solutions randomly.
2. Evaluate the fitness value for each particle.
3. Update $pbest$ and $gbest$ if necessary.
4. Update particle's position according to (5.3).
5. Stop the algorithm if the predefined number of iterations is met; otherwise, go back to step 2

The flowchart corresponding to pseudo code 5.1 is shown in fig. 5.1.

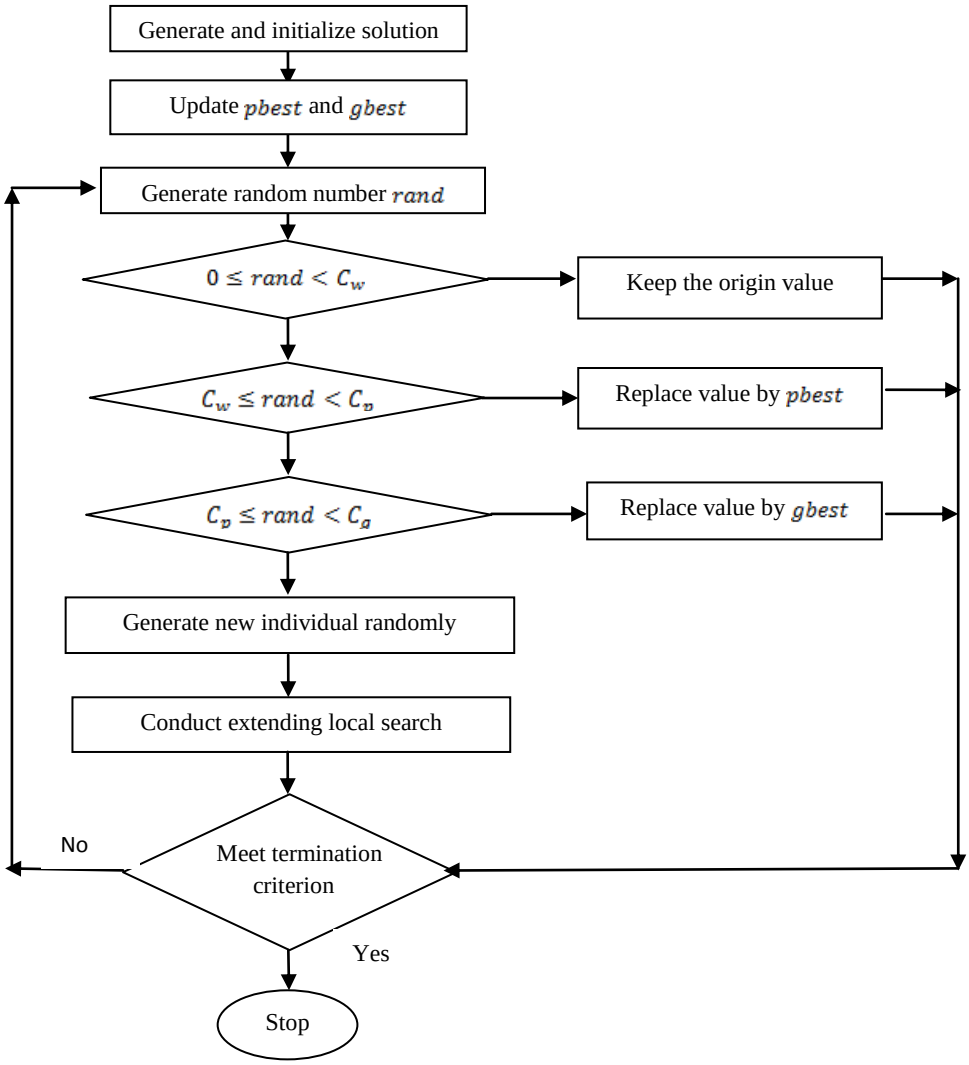


Fig: 5. 1: Flowchart for SSO Algorithm

5.4 Proposed AC-SSO Methodology

AC-SSO is a strong algorithm in solving discrete problems with good global search ability. However, its local search ability is weak compared to PSO. Local search ability is the extent how particles move based on their adjacency. In SSO, Chain Particle (CP) represents the AC, the CP elements are represented as the numbers used in AC and the optimal length of AC i.e., $l(n)$ represents the fitness function, where the initial value of $l(n)$ is computed using eqn. (5.4).

$$\log_2(n) + \log_2(v(n)) - 2.13 \leq l(n) \leq \log_2(n)\,(1 + o(1))/\,(\log_2(n))) \quad \ldots (5.4)$$

where $v(n)$ is Hamming weight. Therefore, $l(2n) \leq l(n) + 1$ ... (5.5)

The main operations of any public-key algorithm is to perform the encryption/decryption operations with modular exponentiation. They involve raising to powers of large field of some group. This process involves multiple multiplications which makes it time consuming. For example, to compute x^e based on the process of adding and multiplying, it needs $(e-1)$ RMs of e ie., $x^1 \rightarrow x^2 \rightarrow x^3 \rightarrow \cdots \rightarrow x^{e-1} \rightarrow x^e$. The optimised AC-SSO is carried out in an iterative way. Optimization of the AC generation begins with small numbers and continues on to large numbers. Optimization is a one-time operation such that it does not affect the runtime for a collection of ACs.

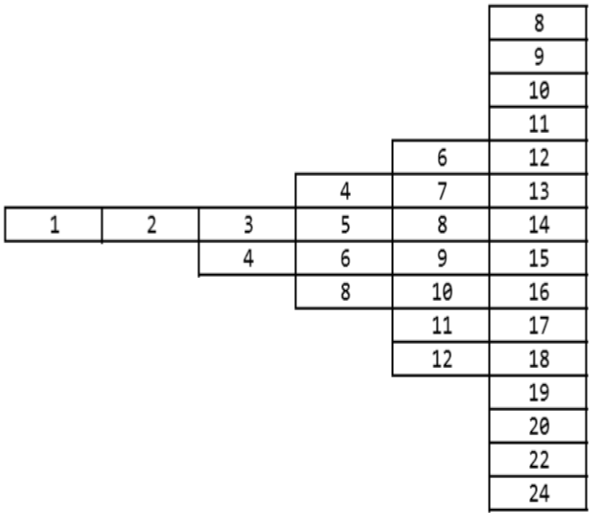


Fig. 5.2: The Chain Particles (CP)

In fig. 5.2, elements with minimum AC duration shall be replaced by particles in SSO. The CP elements are CP_i , $i = 1,2, \ldots n$. The search spaces for the elements are often restricted to simplifying the method of optimization. The first value of the variable is limited to 1 since all ACs should start with 1. The second variable is limited to 2 with a value of 1 doubled. There are no optimization processes involving the first two

elements. A $3(2+1)$ or a 4 may be the third variable $(2X2)4, 5, 6$ or 8 may be the fourth part. After completing all epochs, particle outputs are optimized for SSO particles.

$$CP1 = \{1\}$$

$$CP_2 = \{2\}$$

$$CP_3 = \{3,4\}$$

$$CP_4 = \{4,5,6,8\}$$

$$CP_5 = \{6,7,8,9,10,11,12\}$$

$$CP_6 = \{8,9,19,11,12,13,14,15,16,17,18,19,20,22,24\}$$

There are two possibilities of SSO based ACs achievement in fig. 5.3 and fig.5.4. For example, AC for 78 using binary method is $1-2-4-8-9-18-19-38-39-78$ with length 9.

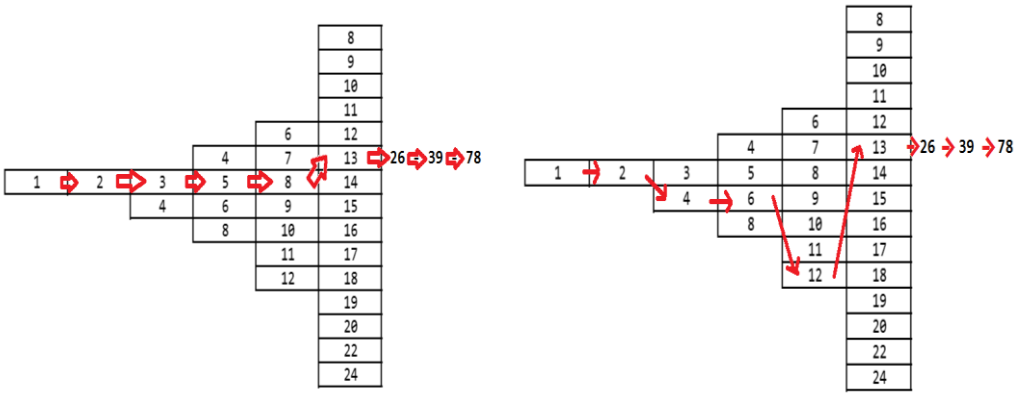


Fig. 5.3 and Fig 5.4 : Two different ACs for the Integer 78 Generated Using SSO

While running SSO optimization, some possible ACs for the value 78 with $l(n) = 8$:

1 2 3 5 8 13 26 39 78| 1 2 3 5 8 13 26 52 78| 1 2 3 5 10 13 26 39 78| 1 2 3 5 10 13 26 52 78| 1 2 3 6 7 13 26 39 78| 1 2 3 6 7 13 26 52 78| 1 2 3 6 9 15 24 39 78| 1 2 3 6 9 15 30 39 78| 1 2 3 6 9 18 21 39 78| 1 2 3 6 9 18 36 39 78| 1 2 3 6 9 18 36 42 78| 1 2 3 6 9 18 36 72 78| 1 2 3 6 12 13 26 39 78| 1 2 3 6 12 13 26 52 78| 1 2 3 6 12 14 26 52 78| 1 2 3 6 12 15 24 39 78| 1 2 3 6 12 15 27 39 78| 1 2 3 6 12 18 21 39 78| 1 2 3 6 12 18 30

48 78| 1 2 3 6 12 18 30 60 78| 1 2 3 6 12 18 36 39 78| 1 2 3 6 12 18 36 42 78| 1 2 3 6 12 18 36 72 78| 1 2 3 6 12 24 26 52 78| 1 2 3 6 12 24 27 39 78| 1 2 3 6 12 24 27 51 78| 1 2 3 6 12 24 27 54 78| 1 2 3 6 12 24 30 48 78| 1 2 3 6 12 24 30 54 78| 1 2 3 6 12 24 36 39 78| 1 2 3 6 12 24 36 42 78| 1 2 3 6 12 24 36 72 78| 1 2 3 6 12 24 48 54 78| 1 2 3 6 12 24 48 72 78| 1 2 4 5 8 13 26 39 78| 1 2 4 5 8 13 26 52 78| 1 2 4 5 9 13 26 39 78| 1 2 4 5 9 13 26 52 78| 1 2 4 6 7 13 26 39 78| 1 2 4 6 7 13 26 52 78 1 2 4 6 10 16 26 52 78| 1 2 4 6 10 20 26 52 78| 1 2 4 6 12 13 26 39 78| 1 2 4 6 12 13 26 52 78| 1 2 4 6 12 14 26 52 78| 1 2 4 6 12 18 30 48 78| 1 2 4 6 12 18 30 60 78| 1 2 4 6 12 18 36 42 78| 1 2 4 6 12 18 36 72 78| 1 2 4 6 12 24 26 52 78| 1 2 4 6 12 24 30 48 78| 1 2 4 6 12 24 30 54 78| 1 2 4 6 12 24 36 42 78| 1 2 4 6 12 24 36 72 78| 1 2 4 6 12 24 48 54 78| 1 2 4 6 12 24 48 72 78| 1 2 4 8 9 13 26 39 78| 1 2 4 8 9 13 26 52 78| 1 2 4 8 9 17 26 52 78| 1 2 4 8 9 18 26 52 78| 1 2 4 8 10 16 26 52 78| 1 2 4 8 10 18 26 52 78| 1 2 4 8 12 13 26 39 78| 1 2 4 8 12 13 26 52 78| 1 2 4 8 12 14 26 52 78| 1 2 4 8 12 24 26 52 78| 1 2 4 8 16 18 26 52 78| 1 2 4 8 16 24 26 52 78|.

After completing all epochs, SSO Optimized particle (element) values are:

$$CP_1 = \{1\}$$
$$CP_2 = \{2\}$$
$$CP_3 = \{3,4\}$$
$$CP_4 = \{5,6,8\}$$
$$CP_5 = \{7,9,10,12,16\}$$
$$CP_6 = \{11,13,14,15,17,18,,20,24,32\}$$

After completing the given epochs, SSO generated optimized particle values in the element positions. The values of SSO optimized result particle are given in fig. 5.5. It starts with feature space and continues on until all the features are visited. While

comparing AC-SSO generation with binary AC generation method, the output of AC-SSO is less. The shorter an exponent is, the less time is needed for its calculation.

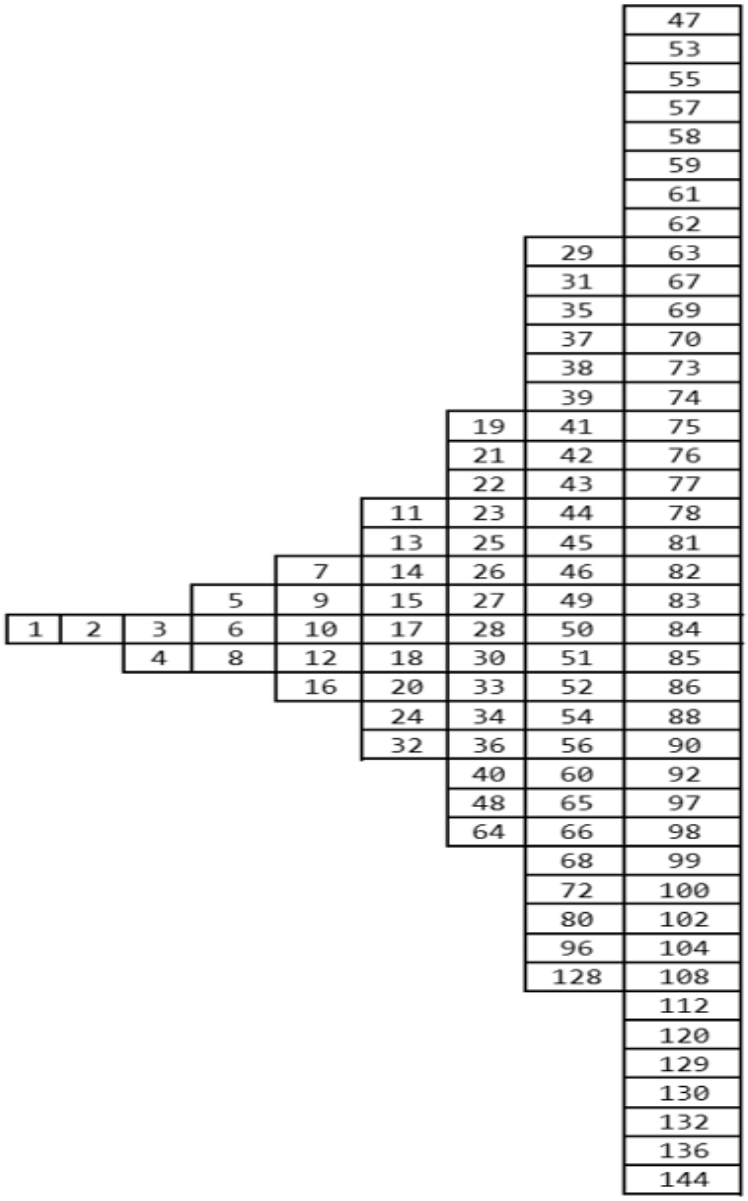


Fig. 5.5: SSO Optimized Result Particle Values

5.5 Results and Discussion

The proposed methodology is implemented in VC++ with Android and Windows emulator for varying file sizes using RSA and ECC. The time taken for ET, DT (in mS), EP, DP (in mW) and SE (in %) are recorded in table from 5.1 to 5.10 and their corresponding graphical representations are shown in fig. 5.6 to 5.15.

Table.5.1: Encryption Time (mS) using AC-SSO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC- SSO based RSA and ECC	
	without RM and RA		with RM and RA			
	ET-RSA-A	ET-ECC-A	ET-RM-RSA-A	ET-RA-ECC-A	ET-AC-SSO-RSA-A	ET-AC-SSO-ECC-A
1	1660	2447	1227	1820	1015	1305
2	3237	4790	2377	3555	2001	2547
4	6494	9553	4788	7173	3997	5122
8	13689	20179	10077	15130	8424	10813
16	27426	40415	20199	30326	16899	21666
Total	52506	77384	38668	58004	32336	41453
Avg.	10501.2	15476.8	7733.6	11600.8	6467.2	8290.6

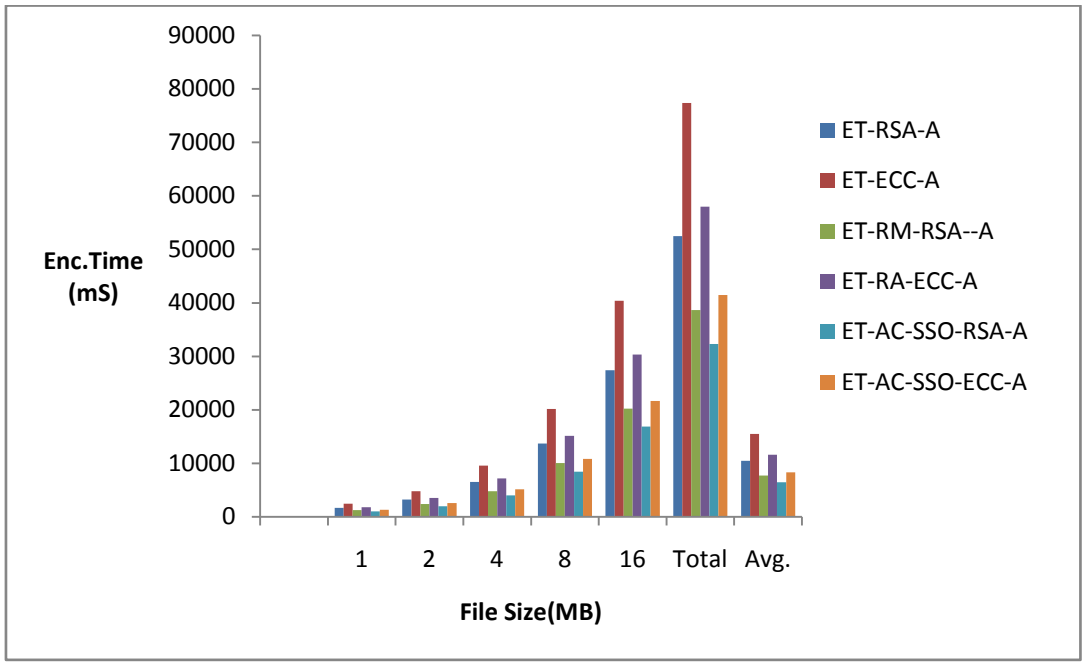


Fig. 5.6: Graph Showing Encryption Time(mS) using AC-SSO in RSA and ECC with Android Emulator

- ET-AC-SSO-RSA-A is 1.624 times faster than ET-RSA-A and 1.196 times faster than ET-RM-RSA-A
- ET-AC- SSO -ECC-A is 1.867 times faster than ET-ECC-A and 1.399 times faster than ET-RA-ECC-A
- ET-AC- SSO -RSA-A is 1.282 times faster than ET-AC- SSO -ECC-A

Table.5.2: Decryption Time (mS) using AC-SSO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC- SSO based RSA and ECC	
	without RM and RA		with RM and RA			
	DT-RSA-A	DT-ECC-A	DT-RM-RSA-A	DT-RA-ECC-A	DT-AC-SSO-RSA-A	DT-AC-SSO-ECC-A
1	1616	2330	1205	1789	1014	1282
2	3193	4519	2352	3519	1974	2533
4	6490	9209	4776	7171	4000	5125
8	13645	19375	10066	15082	8406	10785
16	27399	38936	20193	30294	16877	21646
Total	52343	74369	38592	57855	32271	41371
Avg.	10468.6	14873.8	7718.4	11571	6454.2	8274.2

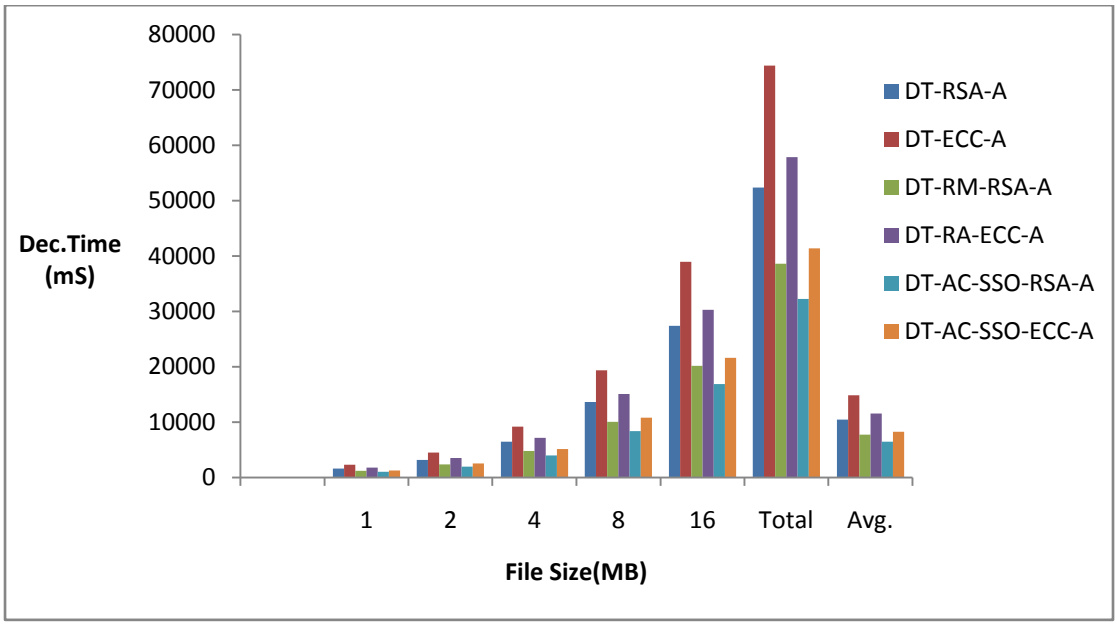


Fig. 5.7: Graph Showing Decryption Time(mS) using AC-SSO in RSA and ECC with Android Emulator

- DT-AC- SSO -RSA-A is 1.622 times faster than DT -RSA-A and 1.196 times faster than DT -RM-RSA-A
- DT -AC- SSO -ECC-A is 1.798 times faster than DT -ECC-A and 1.398 times faster than DT -RA-ECC-A
- DT -AC- SSO -RSA-A is 1.282 times faster than DT -AC- SSO -ECC-A

Table.5.3: Encryption Power (mW) using AC-SSO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC- SSO based RSA and ECC	
	without RM and RA		with RM and RA			
	EP-RSA-A	EP-ECC-A	EP-RM-RSA-A	EP-RA-ECC-A	EP-AC-SSO-RSA-A	EP-AC-SSO-ECC-A
1	554	817	421	613	341	424
2	1102	1621	806	1197	673	792
4	2171	3208	1612	2410	1337	1598
8	4569	6731	3364	5042	2820	3374
16	9156	13510	6735	10134	5633	6752
Total	17552	25887	12938	19396	10804	12940
Avg.	3510.4	5177.4	2587.6	3879.2	2160.8	2588

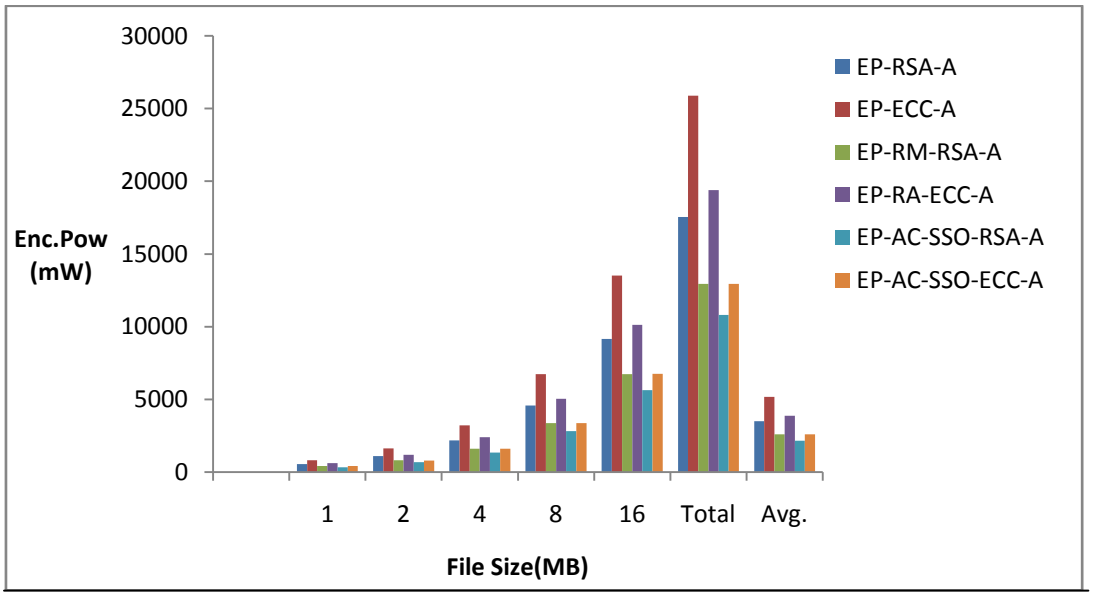


Fig. 5.8: Graph Showing Encryption Power (mW) using AC-SSO in RSA and ECC with Android Emulator

- EP-AC- SSO -RSA-A is 1.625 times less than EP -RSA-A and 1.198 times less than EP -RM-RSA-A.
- EP -AC- SSO -ECC-A is 2.001 times less than EP -ECC-A and 1.499 times less than EP -RA-ECC-A
- EP -AC- SSO -RSA-A is 1.198 times less than EP -AC- SSO -ECC-A

Table.5.4: Decryption Power (mW) using AC-SSO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC- SSO based RSA and ECC	
	without RM and RA		with RM and RA			
	DP-RSA-A	DP-ECC-A	DP-RM-RSA-A	DP-RA-ECC-A	DP-AC-SSO-RSA-A	DP-AC-SSO-ECC-A
1	562	815	411	602	339	399
2	1081	1506	802	1178	669	799
4	2175	3072	1608	2417	1344	1600
8	4548	6463	3357	5035	2815	3361
16	9148	12995	6743	10109	5634	6739
Total	17514	24851	12921	19341	10801	12898
Avg.	3502.8	4970.2	2584.2	3868.2	2160.2	2579.6

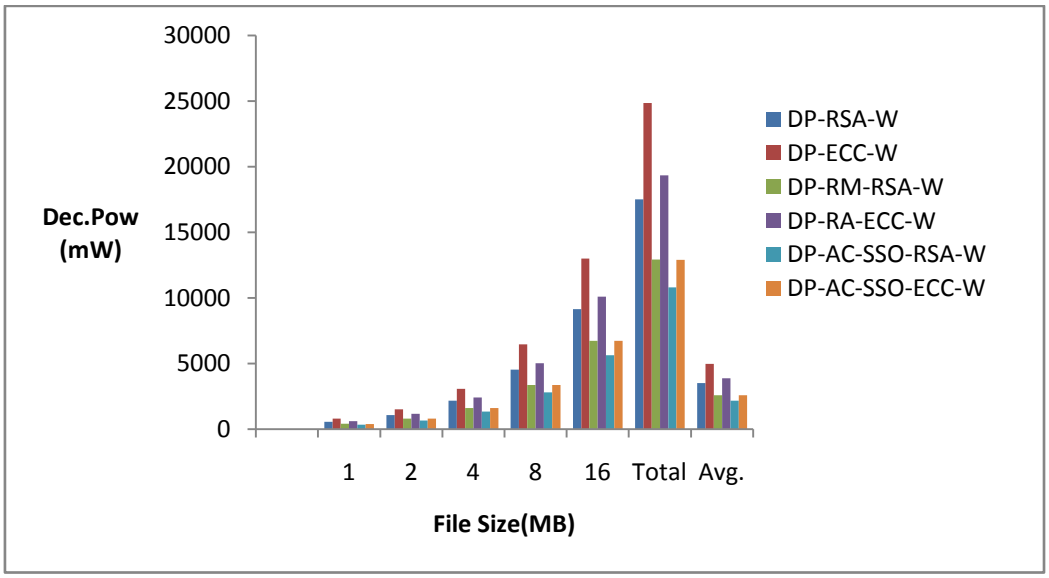


Fig. 5.9: Graph Showing Decryption Power in RSA and ECC with SSO Addition Chain using Android Emulator

- DP-AC- SSO -RSA-A is 1.622 times less than DP -RSA-A and 1.196 times less than EP -RM-RSA-A.
- DP -AC- SSO -ECC-A is 1.926 times less than DP -ECC-A and 1.500 times less than DP -RA-ECC-A
- DP -AC- SSO -RSA-A is 1.194 times less than DP -AC- SSO -ECC-A

Table.5.5: Security (%) using AC-SSO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC- SSO based RSA and ECC	
	without RM and RA		with RM and RA			
	SE-RSA-A	SE-ECC-A	SE-RM-RSA-A	SE-RA-ECC-A	SE-AC-SSO-RSA-A	SE-AC-SSO-ECC-A
1	92	93	92	94	93	95
2	90	91	89	92	92	94
4	88	91	88	90	91	92
8	87	90	87	90	90	92
16	86	88	87	89	89	91
Avg.	88.6	90.6	88.6	91	91	92.8

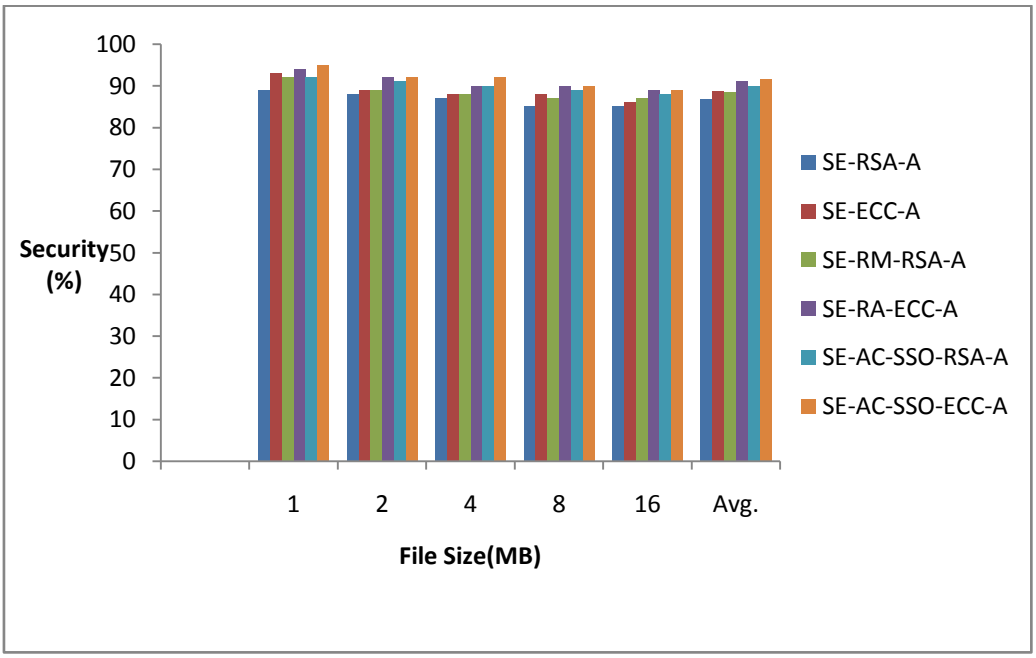


Fig. 5.10: Graph Showing Security (%) using AC-SSO in RSA and ECC with Android Emulator

- SE-AC- SSO-RSA-A is 1.027 times more than SE-RSA-A and 1.027 times more than SE-RM-RSA-A
- SE-AC- SSO -ECC-A is 1.024 times more than SE-ECC-A and 1.020 times more than SE-RA-ECC-A
- SE-AC- SSO -ECC-A 1.020 times more than SE-AC- SSO -RSA-A.

Table.5.6: Encryption Time (mS) using AC-SSO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC- SSO based RSA and ECC	
	without RM and RA		with RM and RA			
	ET-RSA-W	ET-ECC-W	ET-RM-RSA-W	ET-RA-ECC-W	ET-AC-SSO-RSA-W	ET-AC-SSO-ECC-W
1	1664	2441	1204	1728	1009	1299
2	3233	4748	2378	3362	1991	2550
4	6490	9559	4775	6753	3995	5119
8	13670	20174	10082	14252	8427	10819
16	27432	40443	20214	28603	16893	21646
Total	52489	77365	38653	54698	32315	41433
Avg.	10497.8	15473	7730.6	10939.6	6463	8286.6

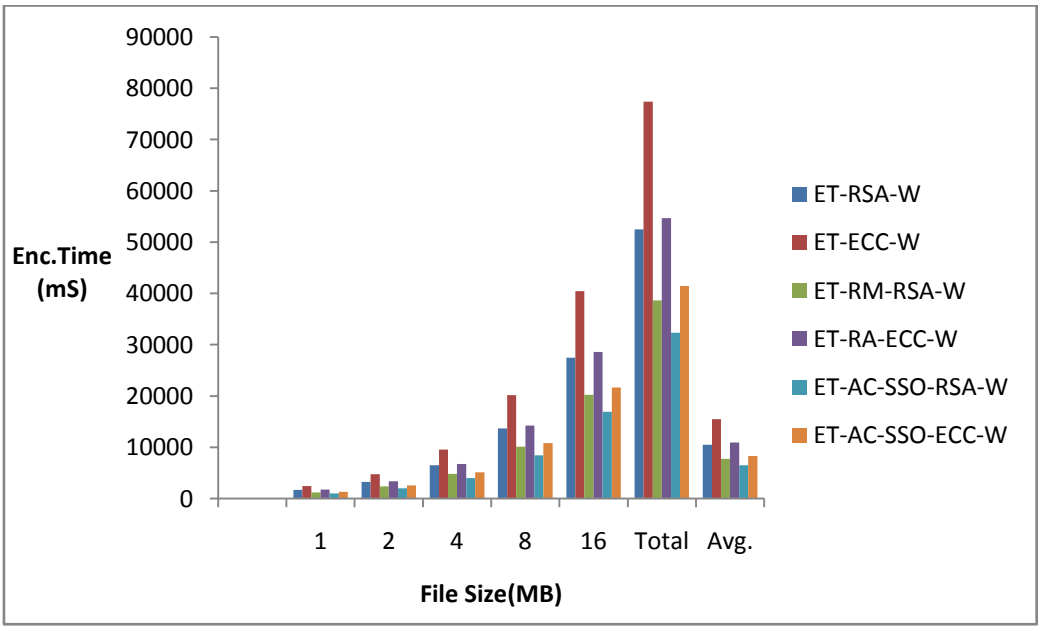


Fig. 5.11: Graph Showing Encryption time in RSA and ECC with SSO Addition Chain using Windows Emulator

- ET-AC- SSO -RSA-W is 1.624 times faster than ET-RSA-W and 1.196 times faster than ET-RM-RSA-W
- ET-AC- SSO -ECC-W is 1.867 times faster than ET-ECC-W and 1.320 times faster than ET-RA-ECC-W
- ET-AC- SSO -RSA-W is 1.282 times faster than ET-AC- SSO -ECC-W

Table.5.7: Decryption Time (mS) using AC-SSO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC- SSO based RSA and ECC	
	without RM and RA		with RM and RA			
	DT-RSA-W	DT-ECC-W	DT-RM-RSA-W	DT-RA-ECC-W	DT-AC-SSO-RSA-W	DT-AC-SSO-ECC-W
1	1616	2322	1205	1708	995	1290
2	3184	4530	2366	3336	1967	2515
4	6496	9212	4778	6755	3990	5127
8	13657	19396	10050	14240	8409	10779
16	27401	38958	20210	28571	16887	21643
Total	52354	74418	38609	54610	32248	41354
Avg.	10470.8	14883.6	7721.8	10922	6449.6	8270.8

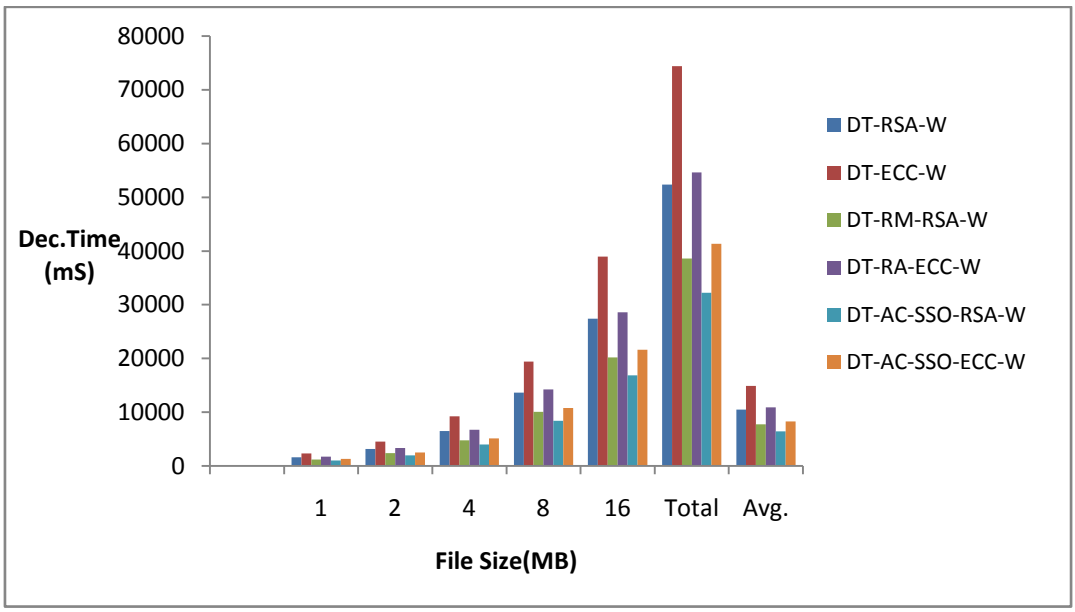


Fig. 5.12 : Graph Showing Decryption Time(mS) using AC-SSO in RSA and ECC with Windows Emulator

- DT-AC- SSO -RSA-W is 1.623 times faster than DT -RSA-W and 1.197 times faster than DT -RM-RSA-W
- DT -AC- SSO -ECC-W is 1.800 times faster than DT -ECC-W and 1.321 times faster than DT -RA-ECC-W
- DT -AC- SSO -RSA-W is 1.282 times faster than DT -AC- SSO -ECC-W

Table.5.8: Encryption Power (mW) using AC-SSO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-SSO based RSA and ECC	
	without RM and RA		with RM and RA			
	EP-RSA-W	EP-ECC-W	EP-RM-RSA-W	EP-RA-ECC-W	EP-AC-SSO-RSA-W	EP-AC-SSO-ECC-W
1	571	840	421	592	345	406
2	1100	1582	796	1146	669	812
4	2179	3228	1604	2263	1341	1597
8	4577	6750	3361	4765	2824	3374
16	9165	13496	6738	9559	5646	6739
Total	17592	25896	12920	18325	10825	12928
Avg.	3518.4	5179.2	2584	3665	2165	2585.6

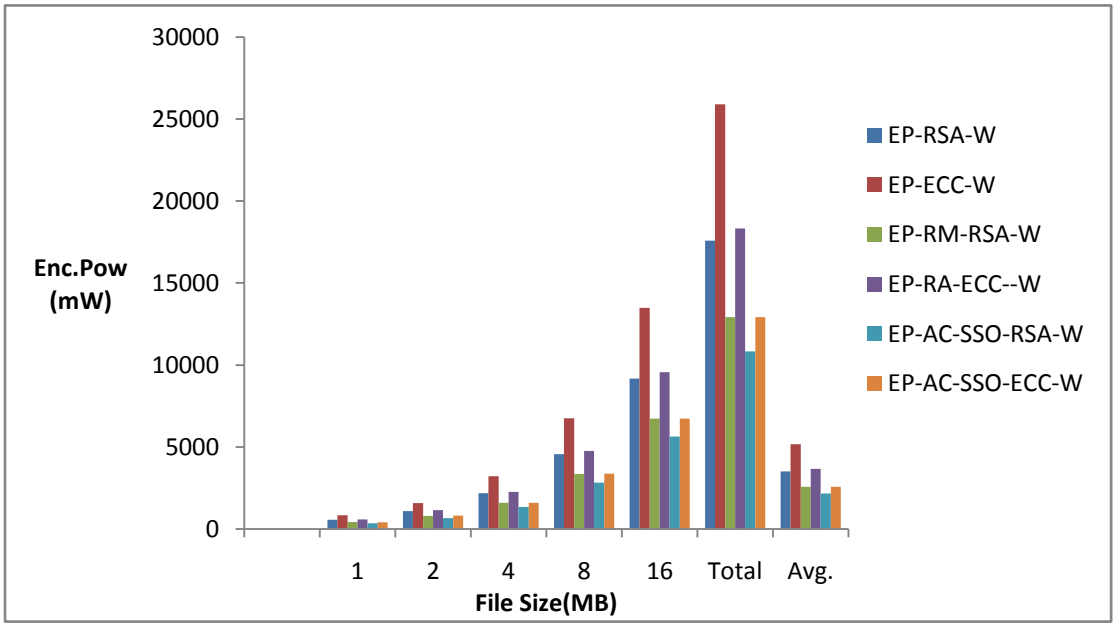


Fig. 5.13: Graph Showing Encryption Power (mW) using AC-SSO in RSA and ECC with Windows Emulator

- EP-AC- SSO -RSA-W is 1.625 times less than EP -RSA-W and 1.194 times less than EP -RM-RSA-W
- EP -AC- SSO -ECC-W is 2.003 times less than EP -ECC-W and 1.417 times less than EP -RA-ECC-W
- EP -AC- SSO -RSA-W is 1.194 times less than EP -AC- SSO -ECC-W

Table.5.9: Decryption Power (mW) using AC-SSO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC- SSO based RSA and ECC	
	without RM and RA		with RM and RA			
	DP-RSA-W	DP-ECC-W	DP-RM-RSA-W	DP-RA-ECC-W	DP-AC-SSO-RSA-W	DP-AC-SSO-ECC-W
1	541	785	415	584	342	420
2	1086	1536	805	1140	655	784
4	2192	3088	1604	2275	1344	1608
8	4579	6496	3368	4761	2819	3360
16	9133	12997	6742	9529	5641	6739
Total	17531	24902	12934	18289	10801	12911
Avg.	3506.2	4980.4	2586.8	3657.8	2160.2	2582.2

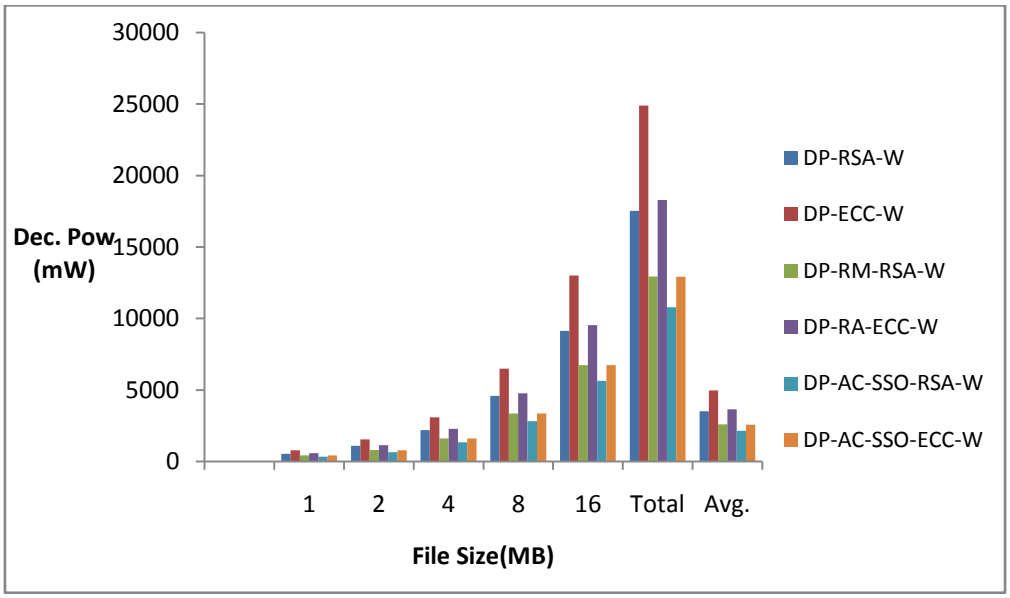


Fig. 5.14: Graph Showing Decryption Power (mW) using AC-SSO in RSA and ECC with Windows Emulator

- DP-AC- SSO -RSA-W is 1.623 times less than DP -RSA-W and 1.197 times less than DP -RM-RSA-W
- DP -AC- SSO -ECC-W is 1.929 times less than DP -ECC-W and 1.417 times less than DP -RA-ECC-W
- DP -AC- SSO -RSA-W is 1.195 times less than DP -AC- SSO -ECC-W

Table.5.10: Security (%) using AC-SSO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC- SSO based RSA and ECC	
	without RM and RA		with RM and RA			
	SE-RSA-W	SE-ECC-W	SE-RM-RSA-W	SE-RA-ECC-W	SE-AC-SSO-RSA-W	SE-AC-SSO-ECC-W
1	91	92	92	94	93	94
2	88	89	89	92	92	93
4	86	89	88	91	90	91
8	86	88	88	89	89	91
16	85	87	87	89	89	89
Avg.	87.2	89	88.8	91	90	91.6

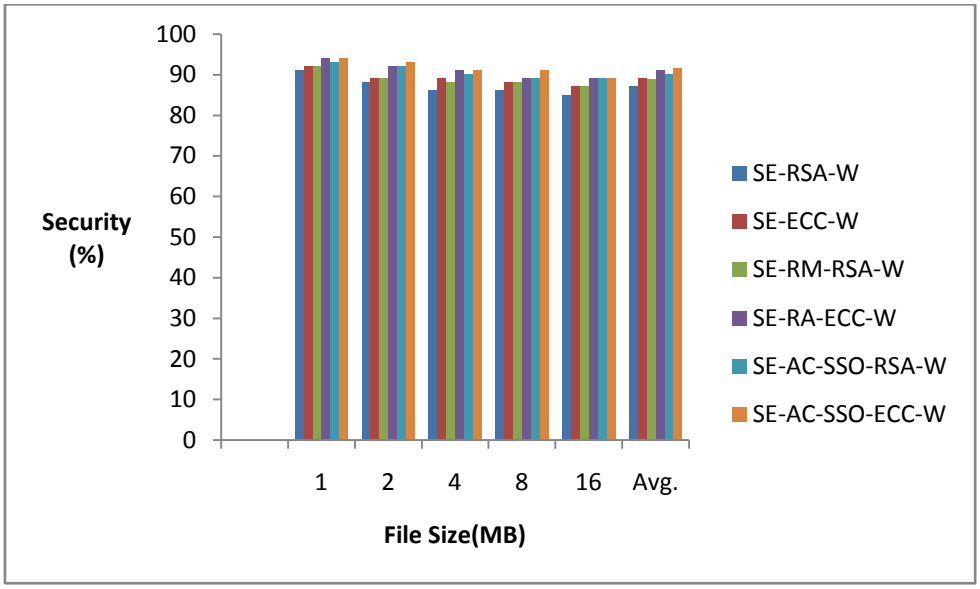


Fig. 5.15: Graph Showing Security (%) using AC-SSO in RSA and ECC with Windows Emulator

- SE-AC- SSO -RSA-W is 1.032 times more than SE-RSA-W and 1.014 times more than SE-RM-RSA-W
- SE-AC- SSO -ECC-W is 1.029 times more than SE-ECC-W and 1.007 times more than SE-RA-ECC-W
- SE-AC- SSO -ECC-W is 1.017 times more than SE-AC- SSO -RSA-W

From the above results, ECC takes more time than RSA for both operational and power consumption. This is because ECC is computationally more intensive approach than RSA. The time required for operational and power consumption when using AC-SSO is substantially reduced compared to without AC.

5.6 Chapter Summary

ACs based on SSO are thought of and incorporated into RSA and ECC. They are implemented successfully. The experimental results show that AC-SSO-RSA takes less operational time, consumes less power than RSA, RM-RSA and ECC, RA-ECC when using both emulators. Higher protection levels are achieved by AC-SSO-ECC while considering security of AC-SSO-RSA. It is also suggested to use AC-SSO-RSA when there is a small power source for a mobile device to run. AC-SSO-ECC offers security of approximately 92% when security parameter is considered. It is concluded that the experimental findings have clearly shown that the proposed AC-SSO with RSA and ECC cryptography systems can be used either to decrease operating power or to achieve enhanced safety levels that are the primary motive of this work. To reduce the time taken for the said parameters further without compromising the optimal length AC for the given integers, an another BIA BFO is considered and it is discussed in next chapter.

CHAPTER - VI

GENERATION OF ADDITION CHAIN USING BACTERIA FORAGING OPTIMIZATION

6.1 Background

In many number theoretic cryptographic algorithms encryption and decryption is of the form $x^n mod\ p$, where x, n and p are integers. Exponentiation is a fundamental operation in computational number theory which normally takes more time than any arithmetic operations. It may be performed by RMs which will reduce the computational time. To reduce the time further, fewer multiplications are performed in computing the same exponentiation operation using AC [90]. The problem of determining correct sequence of multiplications requires in performing modular exponentiation can be elegantly formulated using the concept of AC. To generate the optimal ACs for the given integer, there are several methods exist in literature. But novel Bacteria Foraging Optimization (BFO) algorithm based AC termed as AC-BFO has been proposed in this chapter. Further, the optimal AC generated for an integer using the proposed method has been verified with existing state art of AC method like genetic algorithm, evolutionary programming in this chapter.

6.2 Theoretical Background of Addition Chain

An AC can be thought of as a sequence of integers in which first number is always 1 and last number is always n where n is an integer for which ACs are to be generated. For finite fields, operations such as exponentiations, inversions, or square roots can be performed efficiently by utilizing an optimal AC, the smallest such AC sequence to reach n. In particular, fast exponentiation and inversion are paramount to the performance of scalar point multiplication $k[P]$ where k is a scalar and P is a point in

EC in ECC, pairings in pairing-based cryptosystems and computing isogenies in the quantum-resistant isogeny-based cryptosystems. In order to get the next number, there are two steps normally used in AC. They are addition and doubling steps i.e., to get the next number (intermediate number) in AC, any two previous numbers are added together in addition step where as in doubling step, the current number is multiplied by two [92][93]. To generate the AC for given n, two types of algorithms are normally used viz., deterministic and stochastic or bio inspired.

Generating optimal AC for the given integer is an NP-hard problem because too many optimal ACs are generated for it. For example, different possible optimal ACs for the integer 21 with length i.e., $l(21) = 6$ are:

1-2-3-4-7-14-21	1-2-3-6- 9-15-21	1-2-4-5-10-20-21	1-2-3-6-9-12-21
1-2-3-5-7 -14 -21	1-2-3-6-9-18-21	1-2-4-8-9-12-21	1-2-3-6-9-15-21
1-2-3-5-8-13 -21	1-2-3-6-12-15-21	1-2-4-8- 9-13-21	1-2-3-6-7-14-21
1-2-3-5 -8 -16-21	1-2-3-6-12-18-21	1-2-4-8-9-17-21	1-2-3-6-9-12-21
1-2-3-5-10-11-21	1-2-4-6-7-14-21	1-2-4-8-10-11-21	1-2-3-5-8-16-21
1-2-3-5-10-20-21	1-2-4-8-16-20-21	1-2-4-8-10-20-21	1-2-4-8-12-13-21
1-2-3-6-7-14-21	1-2-4-8-16-17-21	1-2-4-8-12-20-21	

Fig. 6.1: Optimal ACs for the n= 21 with l(21)= 6

This is because 7 can be obtained by adding ($7 = 3 + 4$, $7 = 2 + 5$, $7 = 1 + 6$), 8 can be obtained by adding ($8 = 4 + 4$, $8 = 3 + 5$) etc.

6.3 Bacteria Foraging Optimization

It is one of the optimization and evolutionary algorithms. It was proposed by Kevin M. Passino in 2000 [32][33] and it has been widely accepted as a new nature- inspired optimization algorithm. It is inspired by the social foraging behaviour of *Escherichia coli* i.e., *E.coli* bacteria present in the human intestine and drawn the attention of

many researchers. The underlying biology behind the foraging is locomotion. It is achieved by a set of tensile flagella during the foraging of the real bacteria. Foraging can be modelled as an optimization process where bacterium seeks to maximize the energy obtained per unit time spent during foraging. If the flagella are rotated in the clockwise direction by the bacterium, the flagellum pulls on the cells which results in independent movement of flagella and the bacterium tumbles with lesser numbers of tumbling. Swimming at a very fast rate of bacterium is performed with the flagella moving in the counter clockwise direction.

The foraging strategy of *E.coli* is achieved by four processes viz., chemotaxis, swarming, reproduction and dispersal. Chemotaxis is a process which simulates the movement of *E.coli* cell through swarming and tumbling via flagella. Movement of E.coli bacterium can be performed in two ways viz.,(i) swim for a period of time in the same direction or it may tumble (ii) alternate between swim and tumble for the entire lifetime. In swarming process, a group of E.coli cells arrange themselves in a travelling ring by moving up the nutrient gradient when placed amidst a semisolid matrix with a single nutrient chemo-effecter. The healthy bacteria asexually split into two bacteria, which are then placed in the same location while the least healthy bacteria eventually die in reproduction process. In elimination and dispersal process, gradual or sudden changes in the local environment i.e., significant local rise of temperature or due to unavoidable events all the bacteria in a region are killed or a group is dispersed into new location.

In BFO, generally the bacteria move for a longer distance in a friendly environment. When they got sufficient food, their lengths are increased and they break in the middle to form an exact replica of itself in the presence of suitable environment. The

chemotactic progress may be destroyed and a group of bacteria may move to some other places or some other may be introduced in the swarm of concepts due to the occurrence of sudden environmental changes. This constitutes the event of elimination-dispersal in the real bacterial population, where all the bacteria in a region are killed or a group is dispersed into a new part of the environment.

6.4 Proposed AC-BFO Methodology

In the proposed methodology, the concept of BFO is used to generate the optimum length AC for an integer n which utilizes foraging behaviour of bacteria. In AC-BFO, each bacterium represents the AC, movement of bacterium towards searching the food represents the intermediate numbers to be generated in AC and the fitness function represents the length of AC for the given number n i.e., $l(n)$. In this optimization, a virtual bacterium called search agent is actually one trial solution that moves on the functional surface to find the optimal length AC. The cost or fitness function is computed with minimum length approach based on the nutrient concentration of the immediate environment of the bacterium searching for numbers in AC. Swarming step is not considered for the generation of AC in this method. The notations used in the proposed AC-BFO methodology shown in table 6.1.

In order to generate the AC for any integer n, first number is always 1 and second number is 2, i.e., AC starts with $a_0 = 1$ and $a_1 = 2$ and last number $a_r = n$. Let $(i, k, l) = \{(j, k, l | i = 1,2, \dots, S)$ represents each number in the AC in the population S at the j^{th} chemotactic step, k^{th} reproduction and l^{th} elimination-dispersal step.

Table 6.1: Notations Used in AC-BFO

j	Index for the chemotactic step
k	Index for the reproduction step
i	Index for the elimination-dispersal event
S	Total number of bacterium in the population
d	Dimension of the search space. Here, $d = 1$
S_w	The swimming length
RP_n	Number of reproduction steps
ED_n	Number of elimination-dispersal events
P_{ed}	Elimination-dispersal probability
$C(i)$	Magnitude of the next number in the random direction specified by the tumble

It is noted that initially S is taken as very large for the given n. Too many ACs are generated for n but all ACs generated are not necessarily optimum. Moreover, generation of optimal AC is an NP - hard problem. The prime steps used in BFO related to generating the AC are as follows.

6.4.1 Search Space

Here, search space is considered as 1-dimension ($i.e., d = 1$). As the numbers involved in generating AC for any integer n and the difference between intermediate numbers in ACs are finite, the search space is also finite.

6.4.2 Chemotaxis

The movement of an *E.coli* cell through swimming and tumbling via flagella is simulated by chemotaxis process. When a bacterium meets a favorable environment (rich in nutrients, and noxious free), it will continue swimming in the same direction. When it meets an unfavorable environment, it will tumble, i.e., change its direction**.** In BFO, *E.coli* can swim for a period of time in the same direction or it may tumble and alternate between these two modes of operation for the entire lifetime. It is the

most important step in determining the optimal AC for n. For AC generation, swimming and tumbling represent addition and doubling step respectively. The goal is to move to let the bacterium search for the next number in the AC with minimal step.

6.4.3 Minimum Intermediate Number in AC

It is noted that the number of intermediate numbers between 2 and n should be minimum and it is obtained by minimum number of steps as far as possible so that $l(n)$ could be minimized by considering all the directions (previous numbers) from the current bacterium position (present current number) can be chosen for the next step. Initially bacterium *i* is positioned at number 1, Let $m = 0$ i.e., $a_0 = 1$. From 1, then it should move to 2. Now, $m = m + 1\ i.e., a_1 = 2$, AC $\leftarrow 1 - 2$; $l(\text{AC}) = 1$. From 2, it can move to either 3 *or* 4, Now, $m = m + 1$

$$= 2a_{m-1} \quad \text{...(6.3)}$$

$$= a_m + a_0, >> k \geq 0 \quad \text{... (6.4)}$$

$$AC \leftarrow AC \parallel a_m \quad \text{... (6.5)}$$

$$\text{Now, (AC)} = \text{AC} + 1\ or\ \text{(AC)} = m \quad \text{.... (6.6)}$$

All the intermediate numbers obtained in this step are added to the minimal set $\phi_{min}\ i.e., \phi_{min} = \{a_m\}$. A random intermediate number $\leq a_m$ is chosen from this set and it indicates the direction of movement (i.e., from which AC starts) of bacterium

$$i\Delta(i) = rand\{x_6 \in \phi_{min}\} \quad \text{…(6.7)}$$

Let, (j, k, l) represents i^{th} bacterium with 1- dimensional vector represented as, 1,2, … , S at j^{th} chromatic, k^{th} reproductive and l^{th} elimination-dispersal step. Let, $C(i)$ be the step size which is taken as unity because from the current number in AC,

only one next number in the AC is generated based on previous numbers. Thus, the movement of bacterium may be represented in chemotaxis process as

$$(j+1,k) = \theta^{i}(j+k) + C(i)\frac{\Delta(i)}{\sqrt{\Delta(i)^{T}\Delta(i)}} \quad \ldots (6.8)$$

where Δ indicates a vector in the random direction whose elements are $[1, x]$. The movement of bacterium is explained with tree diagram shown in fig. 6.2.

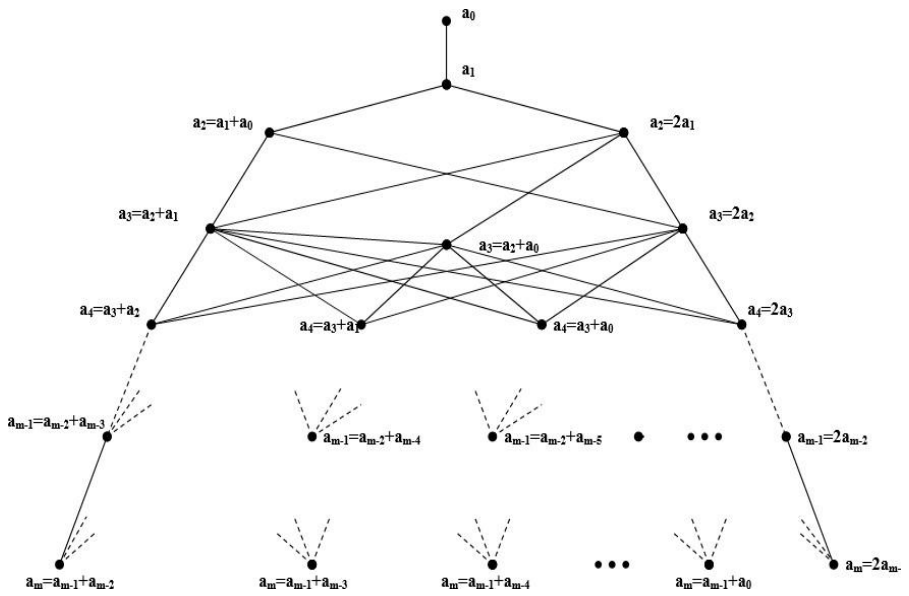


Fig. 6.2: The Movement of Bacterium

6.4.4 Reproduction and Dispersal Step

Local search is provided by chemotaxis step and the speed of convergence is achieved through reproduction process. The bacteria which yields maximal length of AC for n is called least healthy bacteria and it never produces the optimal length AC which eventually dies. Each of the healthiest bacteria (yields minimum length AC) asexually split into two bacteria which are placed in random location. The dispersion process happens after a certain number of reproduction processes. Then, some bacteria are chosen to be killed according to a present probability P_{ed} or moved to another position within the environment. The steps involved in proposed AC-BFO is shown in algorithm 6.1.

Algorithm: 6.1: AC- BFO

1. Initialize parameters $D, S, c_n, R_n, ED_n, \theta^i, C(i), i \leftarrow 2,3, \ldots, S$.

2. Read r // r is the number for which AC is to be found.

3. $m \leftarrow 1;\ a_0 \leftarrow 0, a_1 \leftarrow 1;\ i \leftarrow 0$.

4. $AC^i = a_0 \| a_1$.

5. $(AC^i) \leftarrow 1$ // length of the AC is initialized as 1.

6. $l \leftarrow l + 1$ // Elimination-Dispersal loop.

7. $k \leftarrow k + 1$ // Reproduction loop.

8. $j \leftarrow j + .1$ // Chemotaxis loop.

   8.1. For $i \leftarrow 1,2, \ldots, S$ // Perform the chemotactic step for bacterium i.

   8.2. $m \leftarrow m + 1$.

   8.3. From the initial position of bacterium (i.e. $a0 = 0, a1 = 1$), find all the number of AC^i to reach $a_m \leq r$ with minimum number of steps.

   8.4. Add these numbers in the minimal set ϕ_{min}.

   8.5. Select the number x randomly using eqn. (4). Include x in AC^i. It is also used to find the next number a_m in AC^i.

   8.6. (a) Move: let $(j+1, k) = \theta^i(j + k) + C(i) \frac{\Delta(i)}{\sqrt{\Delta(i)^T \Delta(i)}}$

   (b) Generate a new number a_m for AC^i using eqn.(3).

   $(i, j, k) \leftarrow (i, j-1, k) + a_m$.

   $AC^i \leftarrow AC^i \| ' - ' \| a_m$.

   (c) Compute the length of AC^i (fitness) as $(AC^i) \leftarrow (AC^i) + 1$

   (d) $J_{last} \leftarrow (i, j, k)$.

   (e) if $J(i, j + 1, k) > \{j_{last}\}$ then $r\ J_{last} \leftarrow (i, j + 1, k)$

   else print AC^i, $l(AC^i)$

   (f) update new a_m and AC^i for bacterium i.

9. If $j < C_n$, go to step 8. In this case continue chemotaxis step since the life of the bacterium is not over.

10. Reproduction and elimination.

The reproduction step increases the number of bacteria for better foraging whereas the elimination step removes the bacteria that has traversed all the intermediate numbers.

11. If $k < R_n$, go to step 7. This means that the number of reproduction steps is not reached. So the next generation of the chemotactic loop is started.

12. Selection of optimal length AC

Now, the length of the ACs are generated by each bacterium has been calculated, the AC with minimal length is considered as optimal length AC.

$$l * (AC^i) \leftarrow i^{min} \{l(AC^i)\}$$

The flowchart for the proposed AC-BFO is shown in fig. 6.3.

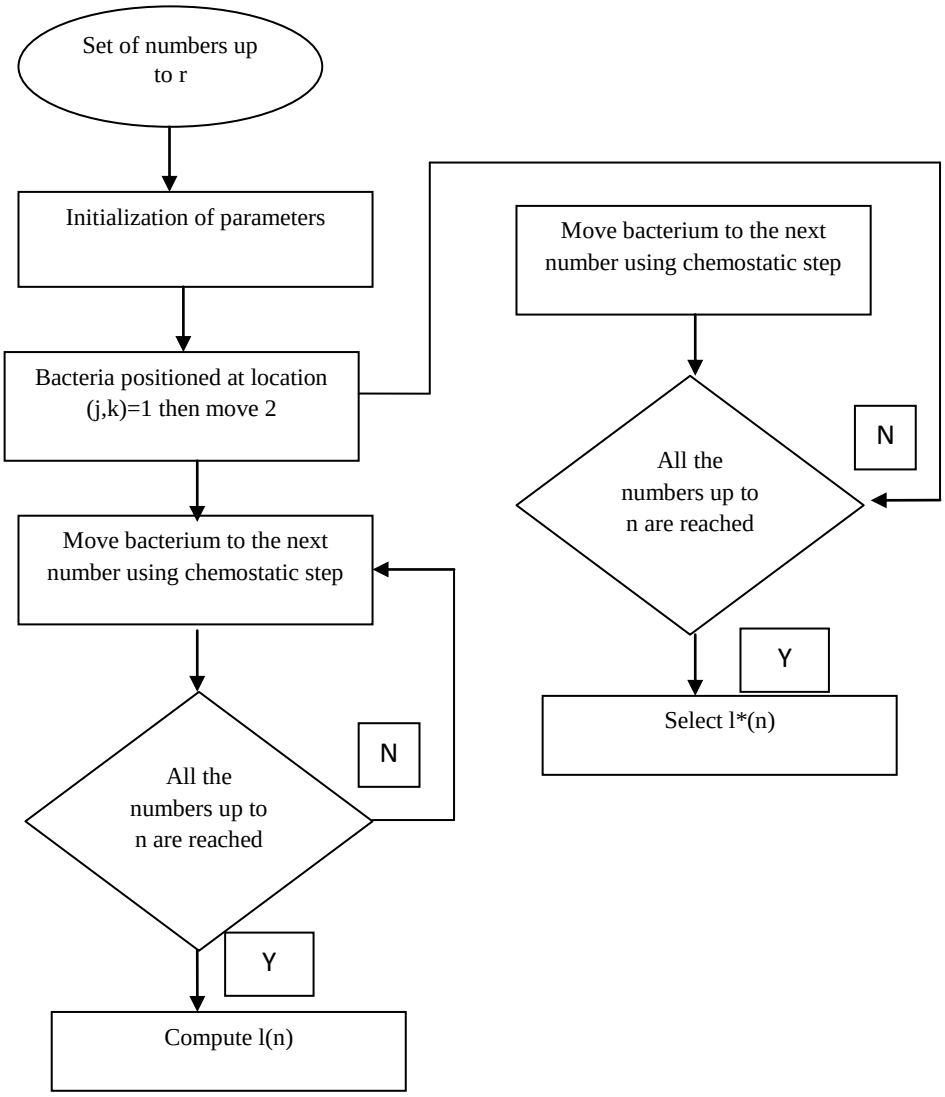


Fig. 6.3: Flowchart for the Proposed AC-BFO

6.4.5 Proposed AC-BFO - An Example

In order to understand the relevance of the work, let, $\mathbf{n = 14, i = 1, m = 0, a_m = a_0 = 1}$ and initially bacteria $\mathbf{b_1}$ is positioned at $\boldsymbol{a_0}$. With the chemotaxis step, it moves to 2. Now, $\boldsymbol{m = m + 1, i.e., \ a_1 = 2 \ and \ l(a_1) = 1}$. From $\mathbf{a_1, b_1}$ moves to either 3 or 4 because $\boldsymbol{a_2 = a_1 + a_0 = 2 + 1 = 3}$ or $\boldsymbol{a_2 = 2a_1 = 4}$. Now, $\mathbf{m = 2}$. Thus $\boldsymbol{\phi_{min}} = \{\mathbf{3, 4}\}$. Let the intermediate number in AC randomly selected from $\mathbf{\Phi_{min}}$, i.e., $\Delta(\mathbf{1}) = \mathbf{3}$.

Thus, the movement of b_1 is from 3, i.e., $a_2 = 3$ and the corresponding AC upto this stage is $1 - 2 - 3$ and $l(a_2) = 2$. From a_2, b_1 moves to either 4 or 5 or 6 because $a_3 = 2a_2 = 6$ or $a_3 = a_2 + a_0 = 3 + 1 = 4$ or $a_3 = a_2 + a_1 = 3 + 2 = 5$. Now, $m = 3$. Thus, $\Phi = \{4,5,6\}$. Let 5 is selected randomly from the set Φ_{min}. Thus, $\Delta(1) = 5$. The movement of b_1 is from 5, $i.e., a_3 = 5$. Correspondingly, AC up to this stage is $1 - 2 - 3 - 5$ and $l(a_3) = 3$. From a_3, b_1 moves to either 6 *or* 7 *or* 8 *or* 10 because $a_4 = 2a_3 = 10 \ or \ a_4 = a_3 + a_0 = 5 + 1 = 6$ or $a_4 = a_3 + a_1 = 5 + 2 = 7$ or $a_4 = a_3 + a_2 = 5 + 3 = 8$. Now, $m = 4$. Thus, $\phi_{min} = \{6,7,8,10\}$. Let 7 is selected randomly from the set ϕ_{min}. Thus, $\Delta(1) = 7$. The movement of 1 is from 7, i.e., $a_4 = 7$. Correspondingly, AC up to this stage is $1 - 2 - 3 - 5 - 7$ and $l(a_4) = 4$. From a_4, b_1 moves to either 8 or 9 or 10 or 12 or 14 because $a_5 = 2, a_4 = 14 \ or \ a_5 = a_4 + a_0 = 7 + 1 = 8 \ or \ a_5 = a_4 + a_2 = 7 + 3 = 10, a_5 = a_4 + a_3 = 7 + 5 = 12$. Now, $m = 5$, Thus, $\phi_{min} = \{8,9,10,12,14\}$. Let 14 is selected randomly from the set ϕ_{min}. Thus, $\Delta(1) = 14$. The process is terminated because it reaches $n = 17$. Correspondingly, AC up to this stage is $1 - 2 - 3 - 5 - 7 - 14$ and $l(a_5) = 5$. Suppose, other numbers from ϕ_{min} is selected, even though it

reaches 14 in the subsequent stages, $l(14)$ is increased and the corresponding bacteria will eventually die. Based on different bacteria movement, the ACs are generated with $l^*(14) = 5$.

1-2-3-4-7-14	1-2-3-6-8-14	1-2-4-5-9-14	1-2-4-6-8-14	1-2-4-8-10-14
1-2-3-5-7-14	1-2-3-6-12-14	1-2-4-5-10-14	1-2-4-6-10-14	1-2-4-8-12-14
1-2-3-6-7-14	1-2-4-5-7-14	1-2-4-6-7-14	1-2-4-6-12-14	

6.5 Proposed AC-BFO-RSA - An Example

In order to understand the relevance of RSA with AC, let p=2957, q=2551, then n = pq = 6012707 and ϕ(n)=(p-1)(q-1)=6007800. Let, the private key e=3674911 because gcd(3674911, 6007800) = 1. Using the extended Euclidean algorithm d is computed as d=422191. The public key is the pair (n=6012707, e=3674911), and the private-key pair is n=6012707,d=422191. To encrypt a message m = 5234673, Then, c= $5234673^{3674911}$ mod 6012707 =3650502. To decrypt C, m= c^d mod n = 3650502^{422191} mod 6012707 = 5234673. It is noted that in the encryption decryption, large numbers are involved which take more time to perform exponentiation operation. To reduce the time, 3674910 and 422190 RMs are required for encryption and decryption respectively when regular RMs are used. To reduce the time further, AC for 3674911 and 422191 are generated usingAC-BFO as follows:

1 - 2 - 3 - 6 - 7 - 14 - 28 - 56 - 112 - 224 - 448 - 896 - 1792 - 1795 - 3587 - 7174 -14348 - 28696 - 57392 - 114784 - 229568 - 459136 - 918272 - 918279 - 1836558 -3673116 – 3674911 and l(3674911) = 26 i.e., it requires only 26 multiplications. Similarly, in the case d = 422191, the AC is 1 - 2 - 3 - 5 - 10 - 20 - 40 - 60 - 63 - 103- 206 - 412 - 824 - 1648 - 3296 - 6592 - 13184 - 26368 - 527336 - 105472 - 105532 -211064 - 422128 – 422191 and l(422191) = 23 i.e., it requires only 23 multiplications.In this way time is substantially reduced.

6.6 Proposed AC-BFO-ECC - An Example

Let the EC and the embedding of K are taken from table 3. Let k= 65131. Then, k[P]= 65131[2252,226996]. Since multiplication operation takes more time than addition, it can be performed by RAs. Thus, it requires 65130 additions. To reduce the number of additions further, again AC is used. Thus, the AC generated for 65131 using AC-PSO is 1- 2 - 3 - 5 - 10 - 20 - 40 - 80 - 120 - 125 - 130- 131 - 250 - 500 - 1000 - 2000 - 4000 - 8000 - 16000 - 32000 - 64000 - 65000 - 65130 – 65131 which requires only 23 additions. Here, 1-2-3-5-…- 65130 – 65131 represents P, 2P, 3P, 5P, …, 65130P and 65131P and they are computed using EC arithmetic as discussed in section 3.7.2.

6.7 Results and Discussion

The proposed methodology is implemented in VC++ with Android and Windows emulators for varying file sizes using RSA and ECC. The time taken for ET, DT (in mS), EP, DP (in mW) and SE (in %) are recorded from Table 6.2 to 6.11 and their corresponding graphical representations are shown in fig. 6.4 to 6.13.

Table 6.2: Encryption Time (mS) using AC-BFO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC- BFO based RSA and ECC	
	without RM and RA		with RM and RA			
	ET-RSA-A	ET-ECC-A	ET-RM-RSA-A	ET-RA-ECC-A	ET-AC-BFO-RSA-A	ET-AC-BFO-ECC-A
1	1660	2447	1227	1820	996	1219
2	3237	4790	2377	3555	1944	2412
4	6494	9553	4788	7173	3895	4849
8	13689	20179	10077	15130	8214	10241
16	27426	40415	20199	30326	16469	20503
Total	52506	77384	38668	58004	31518	39224
Avg.	10501.2	15476.8	7733.6	11600.8	6303.6	7844.8

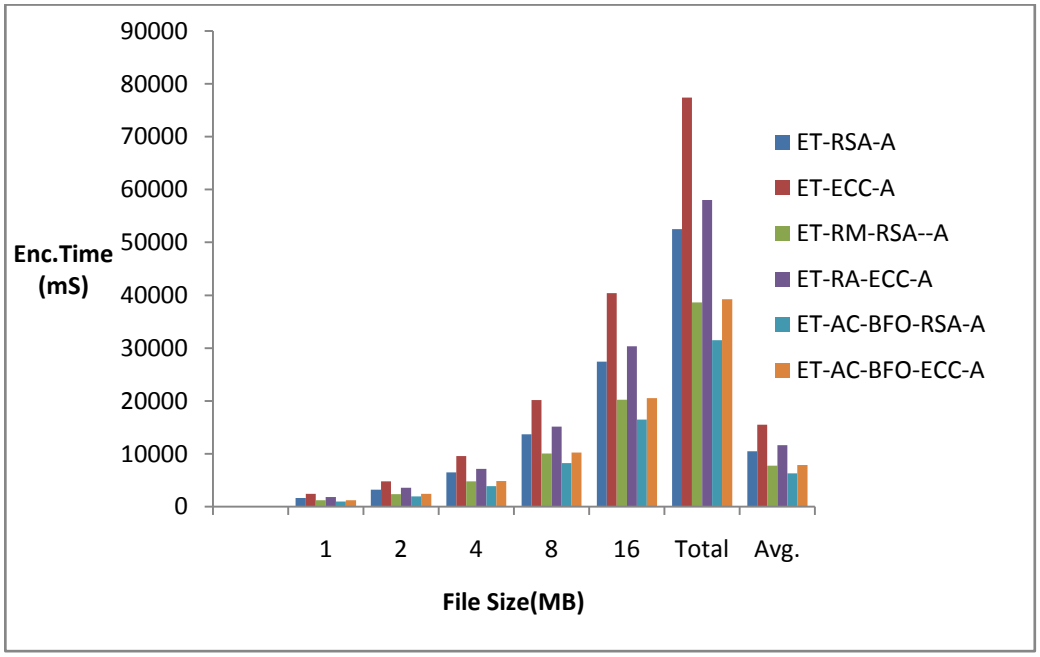


Fig. 6.4: Graph Showing Encryption Time (mS) using AC-BFO in RSA and ECC with Android Emulator

- ET-AC-BFO-RSA-A is 1.666 times faster than ET-RSA-A and 1.227 times faster than ET-RM-RSA-A
- ET-AC- BFO -ECC-A is 1.973 times faster than ET-ECC-A and 1.479 times faster than ET-RA-ECC-A
- ET-AC- BFO -RSA-A is 1.244 times faster than ET-AC- BFO -ECC-A

Table 6.3: Decryption Time (mS) using AC-BFO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC- BFO based RSA and ECC	
	without RM and RA		with RM and RA			
	DT-RSA-A	DT-ECC-A	DT-RM-RSA-A	DT-RA-ECC-A	DT-AC-BFO-RSA-A	DT-AC-BFO-ECC-A
1	1616	2330	1205	1789	972	1224
2	3193	4519	2352	3519	1911	2398
4	6490	9209	4776	7171	3897	4850
8	13645	19375	10066	15082	8185	10208
16	27399	38936	20193	30294	16451	20479
Total	52343	74369	38592	57855	31416	39159
Avg.	10468.6	14873.8	7718.4	11571	6283.2	7831.8

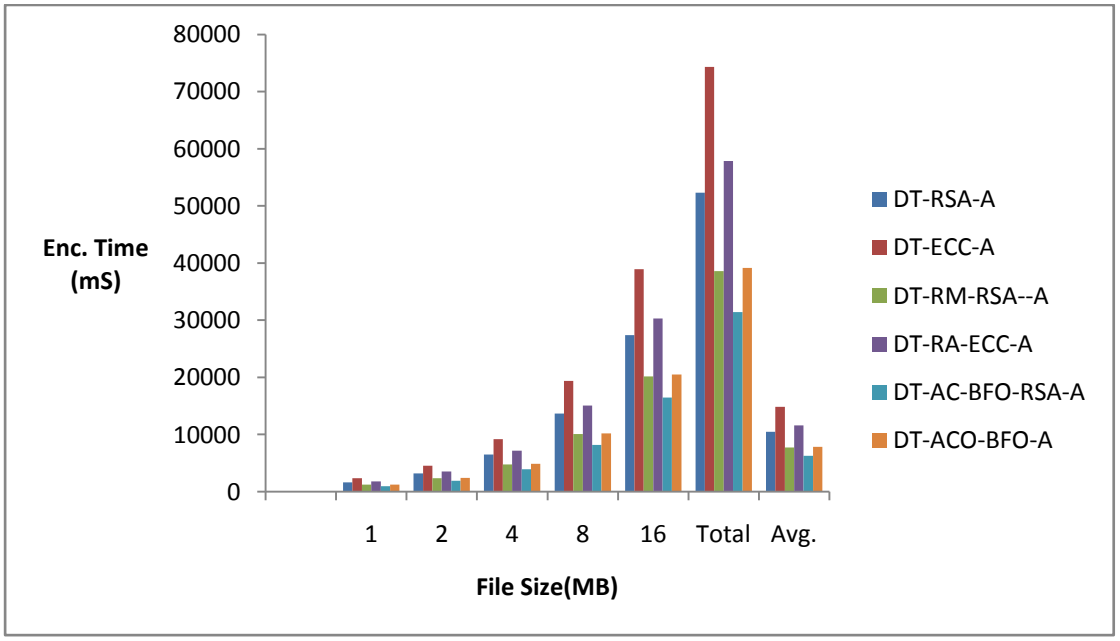


Fig. 6.5: Graph Showing Decryption Time (mS) using AC-BFO in RSA and ECC with Android Emulator

- DT-AC- BFO -RSA-A is 1.666 times faster than DT -RSA-A and 1.288 times faster than DT -RM-RSA-A
- DT -AC- BFO -ECC-A is 1.899 times faster than DT -ECC-A and 1.477 times faster than DT -RA-ECC-A
- DT -AC- BFO -RSA-A is 1.246 times faster than DT -AC- BFO -ECC-A

Table 6.4: Encryption Power (mW) using AC-BFO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC- BFO based RSA and ECC	
	without RM and RA		with RM and RA			
	EP-RSA-A	EP-ECC-A	EP-RM-RSA-A	EP-RA-ECC-A	EP-AC-BFO-RSA-A	EP-AC-BFO-ECC-A
1	554	817	421	613	341	388
2	1102	1621	806	1197	664	776
4	2171	3208	1612	2410	1303	1566
8	4569	6731	3364	5042	2739	3283
16	9156	13510	6735	10134	5494	6557
Total	17552	25887	12938	19396	10541	12570
Avg.	3510.4	5177.4	2587.6	3879.2	2108.2	2514

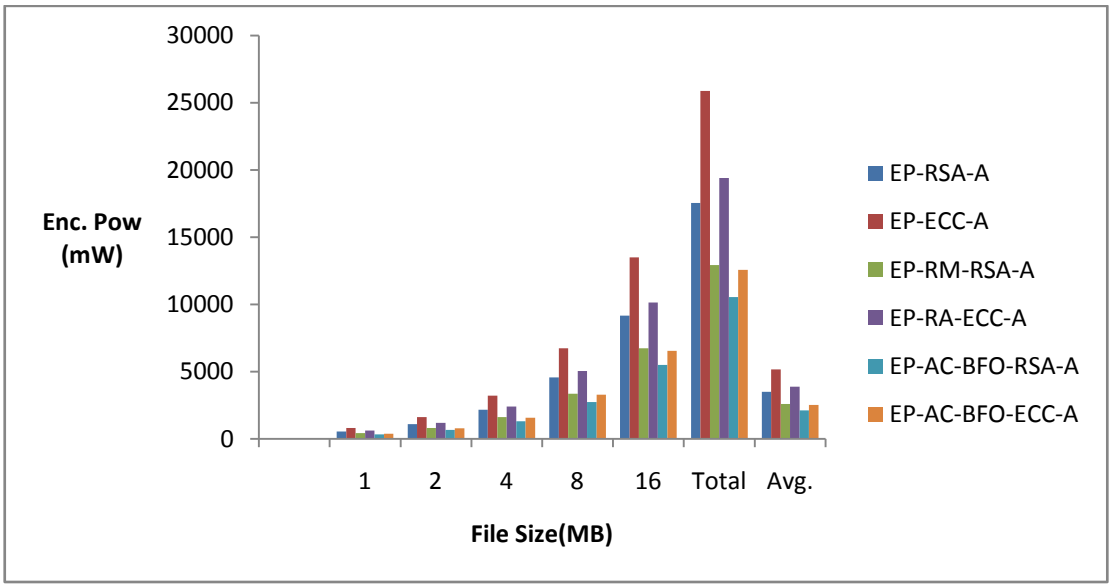


Fig. 6.6: Graph Showing Encryption Power in RSA and ECC with BFO AC Using Android Emulator

- EP-AC- BFO -RSA-A is 1.665 times less than EP -RSA-A and 1.227 times less than EP -RM-RSA-A.
- EP -AC- BFO -ECC-A is 2.059 times less than EP -ECC-A and 1.543 times less than EP -RA-ECC-A
- EP -AC- BFO -RSA-A is 1.192 times less than EP -AC- BFO -ECC-A

Table 6.5: Decryption Power (mW) using AC-BFO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC- BFO based RSA and ECC	
	without RM and RA		with RM and RA			
	DP-RSA-A	DP-ECC-A	DP-RM-RSA-A	DP-RA-ECC-A	DP-AC-BFO-RSA-A	DP-AC-BFO-ECC-A
1	562	815	411	602	334	397
2	1081	1506	802	1178	649	783
4	2175	3072	1608	2417	1299	1561
8	4548	6463	3357	5035	2741	3259
16	9148	12995	6743	10109	5494	6544
Total	17514	24851	12921	19341	10517	12544
Avg.	3502.8	4970.2	2584.2	3868.2	2103.4	2508.8

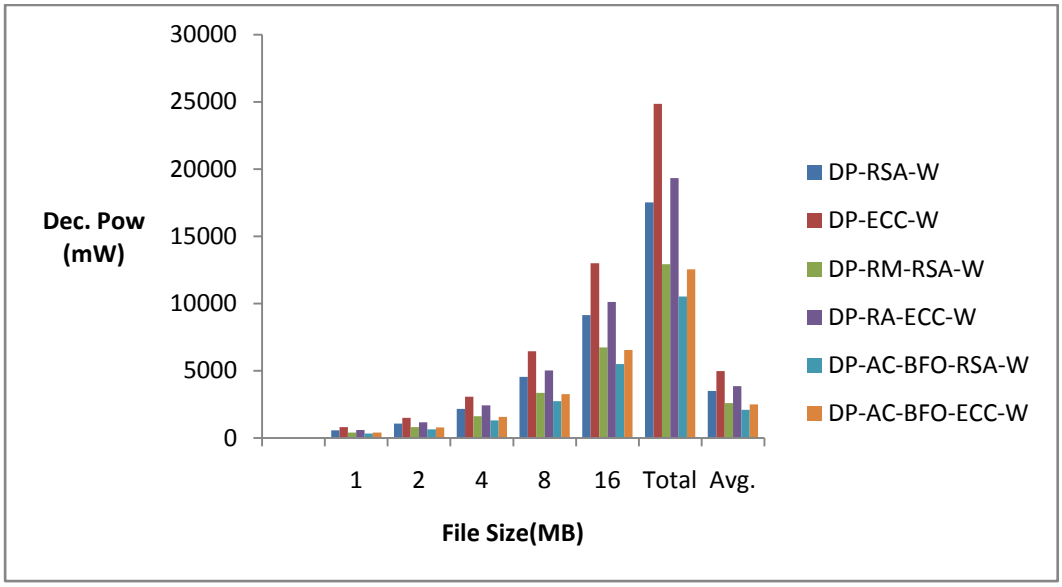


Fig. 6.7: Graph Showing Decryption Power (mW) using AC-BFO in RSA and ECC with Android Emulator

- DP-AC- BFO -RSA-A is 1.665 times less than DP -RSA-A and 1.229 times less than EP -RM-RSA-A.
- DP -AC- BFO -ECC-A is 1.981 times less than DP -ECC-A and 1.542 times less than DP -RA-ECC-A
- DP -AC- BFO -RSA-A is 1.193 times less than DP-AC-BFO -ECC-A

Table 6.6: Security (%) using AC-BFO in RSA and ECC with Android Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-BFO based RSA and ECC	
	without RM and RA		with RM and RA			
	SE-RSA-A	SE-ECC-A	SE-RM-RSA-A	SE-RA-ECC-A	SE-AC-BFO-RSA-A	SE-AC-BFO-ECC-A
1	89	93	92	94	94	96
2	88	89	89	92	91	94
4	87	88	88	90	90	92
8	85	88	87	90	90	91
16	85	86	87	89	89	91
Avg.	86.8	88.8	88.6	91	90.8	92.8

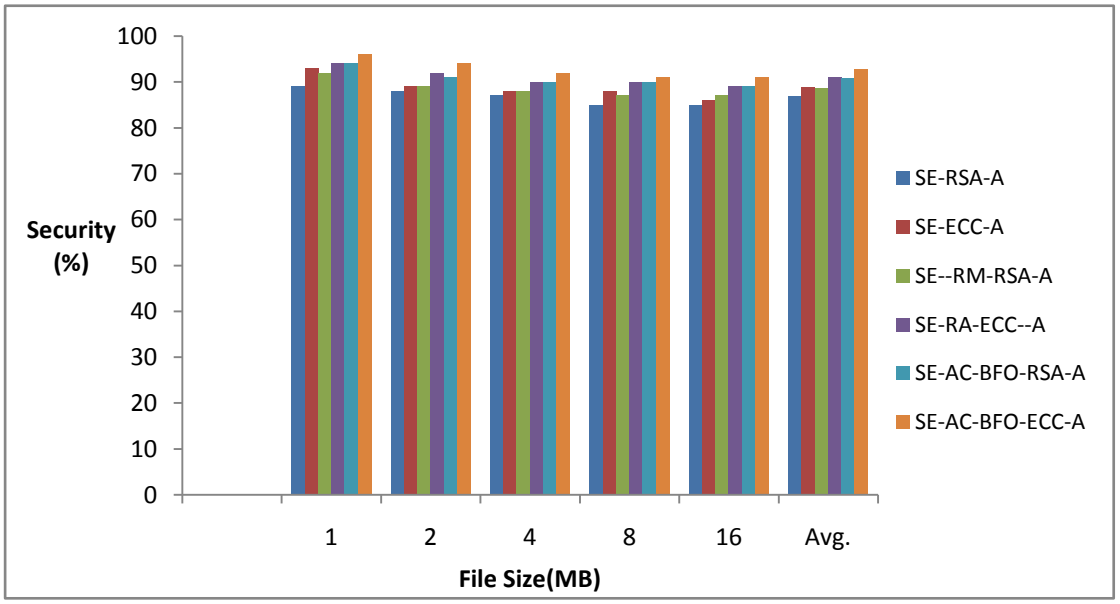


Fig. 6.8: Graph Showing Security (%) using AC-BFO in RSA and ECC with Android Emulator

- SE-AC- BFO-RSA-A is 1.046 times more than SE-RSA-A and 1.025 times more than SE-RM-RSA-A
- SE-AC- BFO -ECC-A is 1.045 times more than SE-ECC-A and 1.020 times more than SE-RA-ECC-A
- SE-AC- BFO -ECC-A is 1.022 times more than SE-AC- BFO -RSA-A

Table 6.7: Encryption Time (mS) using AC-BFO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC- BFO based RSA and ECC	
	without RM and RA		with RM and RA			
	ET-RSA-W	ET-ECC-W	ET-RM-RSA-W	ET-RA-ECC-W	ET-AC-BFO-RSA-W	ET-AC-BFO-ECC-W
1	1654	2441	1204	1728	986	1235
2	3233	4748	2378	3362	1935	2431
4	6490	9559	4775	6753	3888	4846
8	13670	20174	10082	14252	8209	10228
16	27432	40443	20214	28603	16451	20489
Total	52479	77365	38653	54698	31469	39229
Avg.	10495.8	15473	7730.6	10939.6	6293.8	7845.8

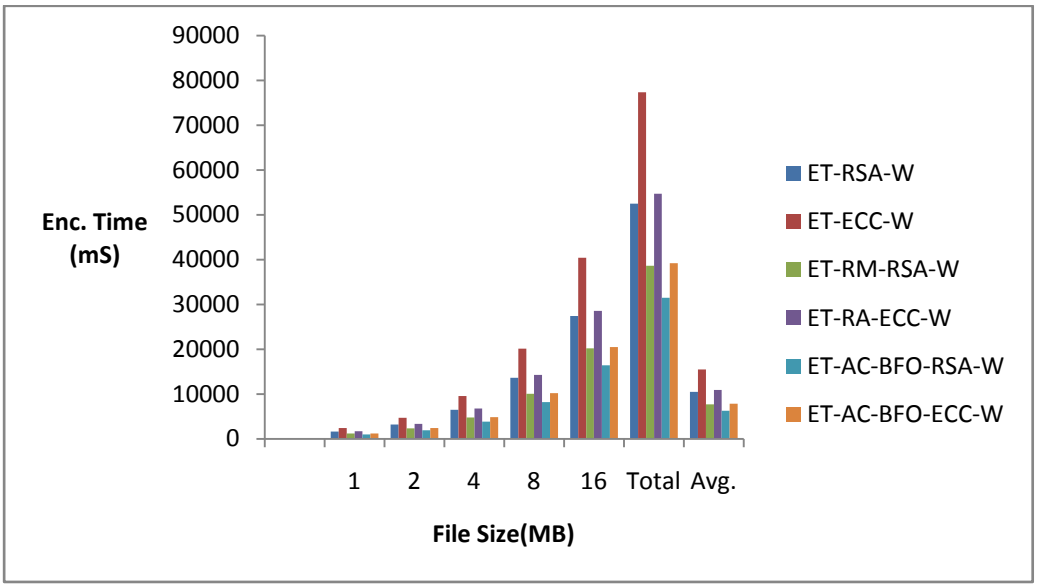


Fig. 6.9: Graph Showing Encryption Time (mS) using AC-BFO in RSA and ECC with Windows Emulator

- ET-AC- BFO -RSA-W is 1.668 times faster than ET-RSA-W and 1.228 times faster than ET-RM-RSA-W
- ET-AC- BFO -ECC-W is 1.972 times faster than ET-ECC-W and 1.394 times faster than ET-RA-ECC-W
- ET-AC- BFO -RSA-W is 1.247 times faster than ET-AC- BFO-ECC-W

Table 6.8: Decryption Time (mS) using AC-BFO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-BFO based RSA and ECC	
	without RM and RA		with RM and RA			
	DT-RSA-W	DT-ECC-W	DT-RM-RSA-W	DT-RA-ECC-W	DT-AC-BFO-RSA-W	DT-AC-BFO-ECC-W
1	1616	2322	1205	1708	998	1226
2	3184	4530	2366	3336	1914	2384
4	6496	9212	4778	6755	3896	4846
8	13657	19396	10050	14240	8187	10192
16	27401	38958	20210	28571	16454	20474
Total	52354	74418	38609	54610	31449	39122
Avg.	10470.8	14883.6	7721.8	10922	6289.8	7824.4

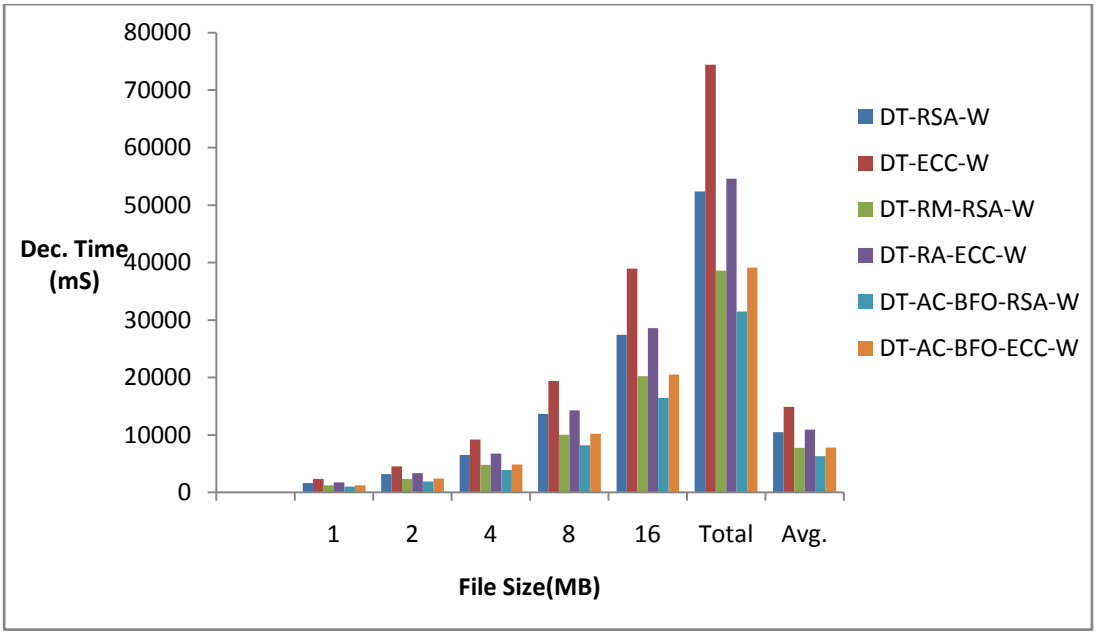


Fig. 6.10: Graph Showing Decryption time in RSA and ECC with BFO AC using Windows Emulator

- DT-AC- BFO -RSA-W is 1.664 times faster than DT -RSA-W and 1.227 times faster than DT -RM-RSA-W
- DT -AC- BFO -ECC-W is 1.902 times faster than DT -ECC-W and 1.395 times faster than DT -RA-ECC-W
- DT -AC-BFO -RSA-W is 1.243 times faster than DT -AC- BFO -ECC-W

Table 6.9: Encryption Power (mW) using AC-BFO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-BFO based RSA and ECC	
	without RM and RA		with RM and RA			
	EP-RSA-W	EP-ECC-W	EP-RM-RSA-W	EP-RA-ECC-W	EP-AC-BFO-RSA-W	EP-AC-BFO-ECC-W
1	571	840	421	592	337	410
2	1100	1582	796	1146	662	779
4	2179	3228	1604	2263	1303	1549
8	4577	6750	3361	4765	2745	3270
16	9165	13496	6738	9559	5494	6541
Total	17592	25896	12920	18325	10541	12549
Avg.	3518.4	5179.2	2584	3665	2108.2	2509.8

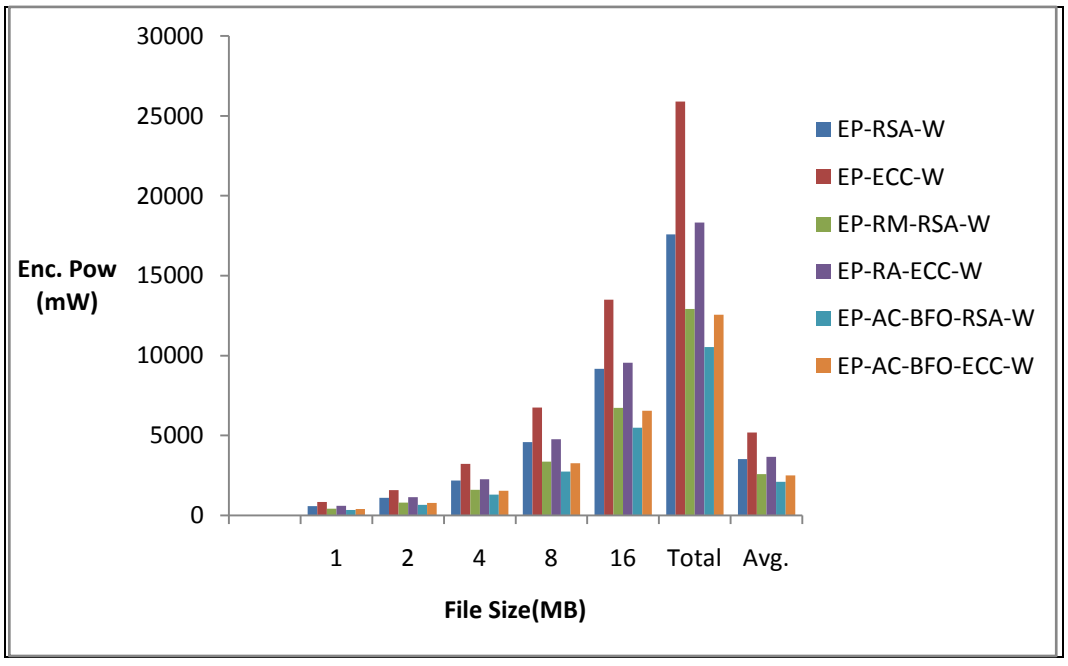


Fig. 6.11: Graph Showing Encryption Power (mW) using AC-BFO in RSA and ECC with Windows Emulator

- EP-AC- BFO -RSA-W is 1.699 times less than EP -RSA-W and 1.226 times less than EP -RM-RSA-W
- EP -AC- BFO -ECC-W is 2.064 times less than EP -ECC-W and 1.460 times less than EP -RA-ECC-W
- EP -AC- BFO-RSA-W is 1.190 times less than EP -AC- BFO-ECC-W

Table 6.10: Decryption Power (mW) using AC-BFO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-BFO based RSA and ECC	
	without RM and RA		with RM and RA			
	DP-RSA-W	DP-ECC-W	DP-RM-RSA-W	DP-RA-ECC-W	DP-AC-BFO-RSA-W	DP-AC-BFO-ECC-W
1	541	785	415	584	332	410
2	1086	1536	805	1140	641	765
4	2192	3088	1604	2275	1312	1558
8	4579	6496	3368	4761	2741	3253
16	9133	12997	6742	9529	5491	6540
Total	17531	24902	12934	18289	10517	12526
Avg.	3506.2	4980.4	2586.8	3657.8	2103.4	2505.2

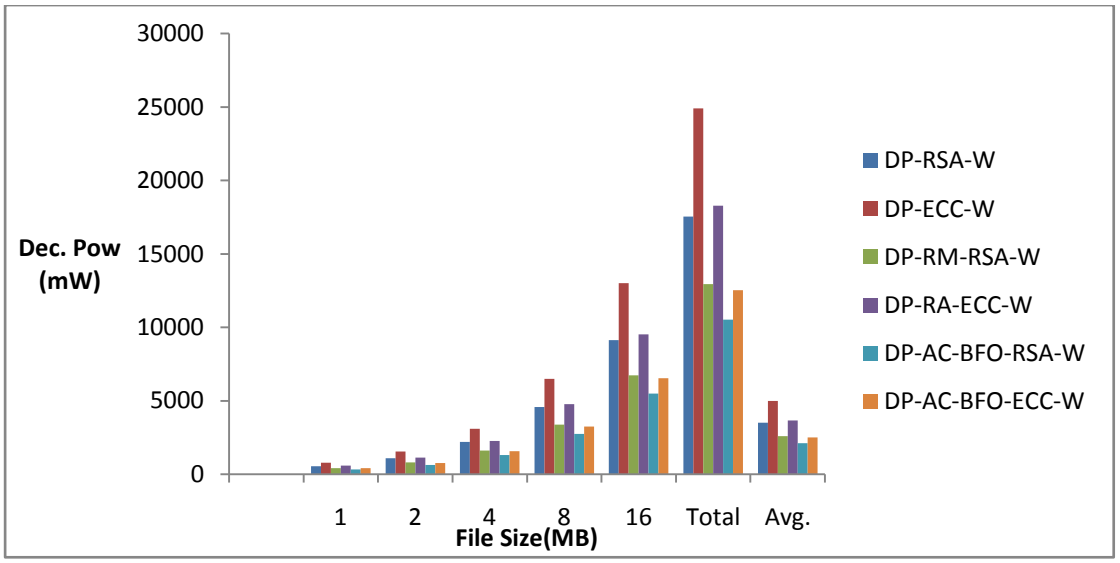


Fig. 6.12 : Graph Showing Decryption Power (mW) using AC-BFO in RSA and ECC with Windows Emulator

- DP-AC- BFO -RSA-W is 1.667 times less than DP -RSA-W and 1.230 times less than DP -RM-RSA-W
- DP -AC- BFO -ECC-W is 1.988 times less than DP -ECC-W and 1.460 times less than DP -RA-ECC-W
- DP -AC-BFO -RSA-W is 1.191 times less than DP -AC-BFO-ECC-W

Table 6.11: Security (%) using AC-BFO in RSA and ECC with Windows Emulator

File Size (MB)	Existing RSA and ECC				Proposed AC-BFO based RSA and ECC	
	without RM and RA		with RM and RA			
	SE-RSA-W	SE-ECC-W	SE-RM-RSA-W	SE-RA-ECC-W	SE-AC-BFO-RSA-W	SE-AC-BFO-ECC-W
1	91	92	92	94	95	96
2	88	89	89	92	91	94
4	86	89	88	91	91	92
8	86	88	88	89	89	91
16	85	87	87	89	88	91
Avg.	87.2	89	88.8	91	90.8	92.8

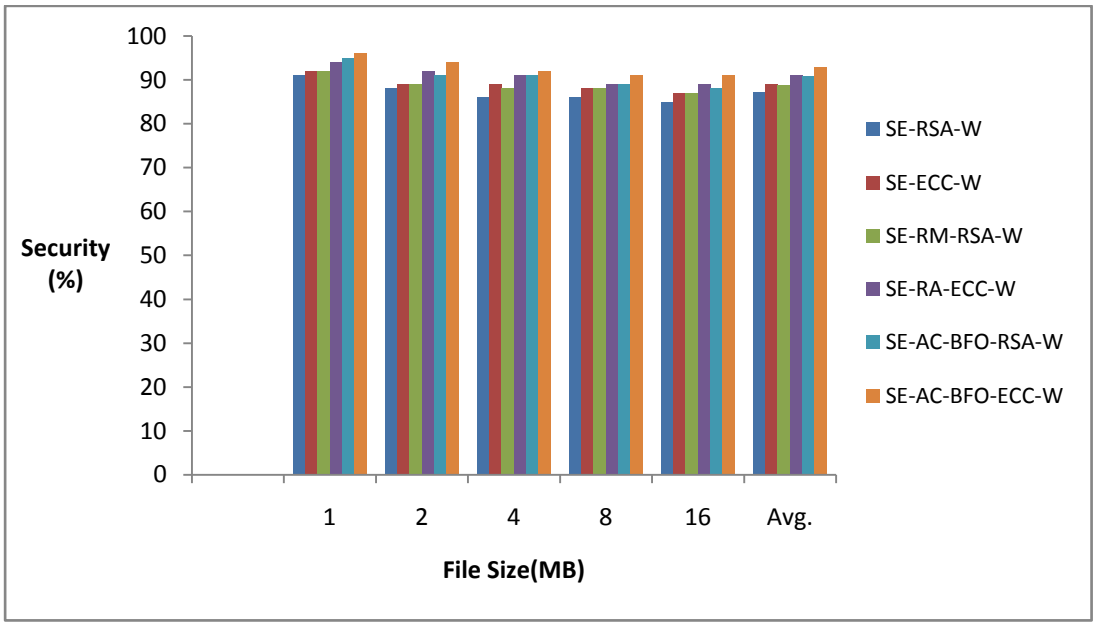


Fig. 6.13: Graph showing the Security (%) using AC-BFO in RSA and ECC with Windows Emulator

- SE-AC- BFO -RSA-W is 1.041 times more than SE-RSA-W and 1.023 times more than SE-RM-RSA-W
- SE-AC- BFO -ECC-W is 1.043 times more than SE-ECC-W and 1.020 times more than SE-RA-ECC-W
- SE-AC- BFO -ECC-W is 1.022 times more than SE-AC-BFO-RSA-W

Table 6.12 shows the generation of AC for some hard exponents where the hard exponent is the one for which AC is not easily generated.

Table 6.12: AC Generated for Some Hard Exponents Using AC-PSO

Integer (n)	**Optimal AC**	$l(n)$
2000	1 – 2 – 3 – 6 – 7 – 14 – 15 – 30 – 31 – 62 – 124 – 125 – 250 – 500 – 1000 – 2000.	15
2048	1 – 2 – 4 – 8 – 16 – 32 – 64 – 128 – 256 – 512 – 1024 - 2048.	11
4096	1 – 2 – 4 – 8 – 16 – 32 – 64 – 128 – 256 – 512 – 1024 – 2048 - 4096.	12
65131	1- 2 - 3 - 5 -10-20-40-80-120-130-250-500-1000-2000-4000-8000-16000--32000-64000-65000-65130-65131	21
196591	1 - 2 - 3 - 5 -10 - 20-30-60-90-180-360-720-1440-1530-3060-6120-12240- 24480-24570-49140-98280-196560-196590-1966591	23
1176431	1-2-3-6-7-14-28-56-112-224-448-896-899-1792-3584-4483-8966-17932-35864-36763-73526-147052-294104-588208-588215-1176430-1176431	26
2211837	1-2-3-6-12-13-26-52-104-208-416-832-1664-3328-3331-6662-9993-16655-33310-66620-133240-266480-276473-5529466-552959-1105918-2211836-2211837	27
4169527	1-2-3-6-9-18-36-45-81-162-243-486-972-1017-2034-4068-8136-16272-32544-65088-130176-260352-260595-521190-1042380-20847660-2084763-41695266-41669527	28
14143037	1-2-3-5-10-20-40-43-86-172-344-688-1376-1379-2758-5516-6695-13790-27580-55160-110320-220640-441280-882560-883939-1767878-3535756-3535759-7071518-141430366-14143037	30

From table 6.13, it is observed that the total length of optimal AC produced by BFO for integers up to 1024 is 11119. They are almost same as the optimal ACs and their length produced by EP.

Table 6.13 : Comparison of AC upto Integers 1024 - Produced by Existing Algorithms and the Proposed AC-BFO

Range of Integers (R)	Opt.	AIS	GA	EP	BFO
[1,512]	4924	4924(+)	4924	4924	4924
[1,1000]	10808	10813(+)	10813	10808	10812
[1,1024]	11115	11120(+)	-	11115	11119

Opt - Optimal AIS - Artificial Immune System EP - Evolutionary Programming GA - Genetic Algorithm

6.8 Chapter Summary

BFO based AC has been thought of and it is implemented successfully. Optimal AC produced by some integers are proved by both theoretically and experimentally. It is observed from the experimental results that upto integers 1024, the proposed AC-BFO produces the same optimal length AC which are exactly equal to other existing evolutional algorithms like AIS and EP. Further, the optimal length of AC for some hard exponents are same as other existing evolutionary methods. The experimental results show that the AC-BFO-RSA takes less operational time, consumes less power than RSA, RM-RSA and ECC, RA-ECC when using both emulators. Higher protection levels are achieved by AC-BFO-ECC while considering security of AC-BFO-RSA. It is also suggested to use AC-BFO-RSA when there is a small power source for a mobile device to run. AC-BFO-ECC offers security of almost 93% when security parameter with windows emulator is considered. It is found that the experimental findings have clearly shown that the proposed AC-BFO with ECC and RSA cryptosystems can be used to enhance the security or to decrease the operational time. As the operational time gets reduced which result in decreasing the operational power. The comparison of all the proposed models on the basis of ET, DT, EP, DP and SE in both android and window emulators are analyzed in next chapter.

CHAPTER - VII

COMPARISON OF PROPOSED BIO-INSPIRED ALGORITHMS FOR ADDITION CHAIN GENERATION WITH RSA AND ECC

From the previous chapters, it is observed that ACs generated by all the proposed bio-inspired algorithms viz., PSO, SSO and BFO take less operational time, less power consumption and more security when they are compared with regular RSA, ECC, RM-RSA and RA-ECC. This chapter provides the comparison of all five parameters values obtained when the existing and the proposed algorithms are implemented with different files sizes 1MB, 2MB, 4MB, 8 MB and 16 MB in A and W emulators.

Table 7.1 to 7.10 show the time taken for ET, DT, EP, DP and SE while using A and W emulator respectively and their graphical representations are shown from fig.7.1 to 7.10.

Table 7.1: Encryption time using AC-PSO-RSA, AC-SSO-RSA &AC-BFO-RSA with Android Vs Windows Emulator

FILE SIZE (MB)	ET-AC-PSO-RSA-A (mS)	ET- AC-PSO-RSA-W (mS)	ET-AC-SSO-RSA-A (mS)	ET- AC-SSO-RSA-W (mS)	ET-AC-BFO-RSA-A (mS)	ET- AC-BFO-RSA-W (mS)
1	1048	1035	866	865	996	994
2	2053	2008	1694	1673	1949	1950
4	4100	4094	3415	3417	3888	3894
8	8643	8624	7211	7185	8220	8213
16	17326	17310	14436	14424	16459	16452
Total	33170	33071	27622	27564	31512	31503
Avg.	6634	6614	5524.4	5513	6302.4	6301

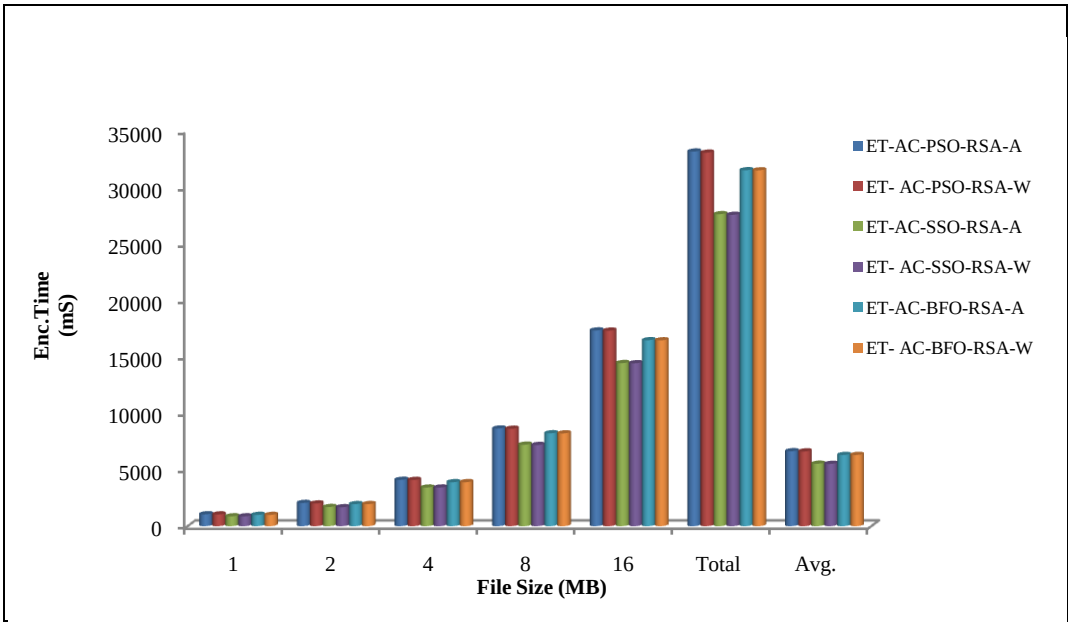


Fig. 7.1: Graph showing the Encryption time using AC-PSO-RSA, AC-SSO-RSA &AC-BFO-RSA with Android Vs Windows Emulator

- ET-AC-PSO-RSA-A is 1.003 times faster than ET- AC-PSO-RSA-W
- ET-AC-SSO-RSA-A is 1.002 times faster than ET- AC-SSO-RSA-W
- ET-AC-BFO-RSA-A is almost same as ET- AC-BFO-RSA-W

It is found that AC based on PSO, SSO and BFO improves the performance of ET when the said methods are used in RSA with A and W emulators. Further, ET-AC-RSA-W takes less ET when they are compared with other proposed methods.

Table 7.2: Encryption time using AC-PSO-ECC, AC-SSO-ECC &AC-BFO-ECC with Android Vs Windows Emulator

FILE SIZE (MB)	ET-AC-PSO-ECC-A (mS)	ET- AC-PSO-ECC-W (mS)	ET-AC-SSO-ECC-A (mS)	ET- AC-SSO-ECC-W (mS)	ET-AC-BFO-ECC-A (mS)	ET- AC-BFO-ECC-W (mS)
1	1571	1533	1299	1302	1222	1228
2	3063	3020	2542	2520	2414	2415
4	6139	6145	5119	5125	4843	4853
8	12970	12918	10819	10767	10221	10232
16	26001	25970	21666	21627	20511	20489
Total	49744	49586	41445	41341	39211	39217
Avg.	9948.8	9917	8289	8268	7842.2	7843

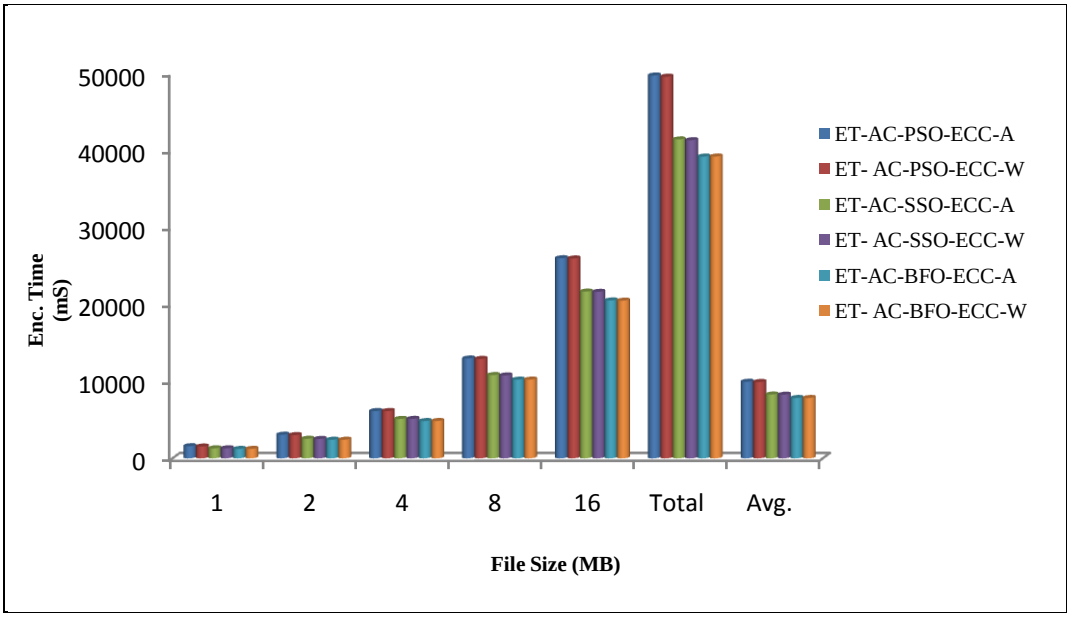


Fig. 7.2: Graph showing the Encryption time using AC-PSO-ECC, AC-SSO-ECC &AC-BFO-ECC with Android Vs Windows Emulator

- ET-AC-PSO-ECC-A is 1.003 times faster than ET- AC-PSO-ECC-W
- ET-AC-SSO-ECC-A is 1.003 times faster than ET- AC-SSO-ECC
- ET-AC-BFO-ECC-A is almost same as ET- AC-BFO-ECC-W

It is also found that AC based on PSO, SSO and BFO improves the performance of ET when the said methods are used in ECC with A and W emulators. Further, ET-AC-ECC-A and ET-AC-ECC-W take less ET when they are compared with other proposed methods.

Table 7.3: Decryption time using AC-PSO-RSA, AC-SSO-RSA &AC-BFO-RSA with Android Vs Windows Emulator

FILE SIZE (MB)	DT-AC-PSO-RSA-A (mS)	DT- AC-PSO-RSA-W (mS)	DT-AC-SSO-RSA-A (mS)	DT- AC-SSO-RSA-W (mS)	DT-AC-BFO-RSA-A (mS)	DT- AC-BFO-RSA-W (mS)
1	363	352	302	293	987	967
2	692	673	573	562	1918	1909
4	1381	1374	1141	1144	3897	3888
8	2896	2881	2418	2398	8182	8190
16	5781	5781	4827	4812	16458	16444
Total	11113	11061	9261	9209	31442	31398
Avg.	2222.6	2212.2	1852.2	1841.8	6288.4	6279.6

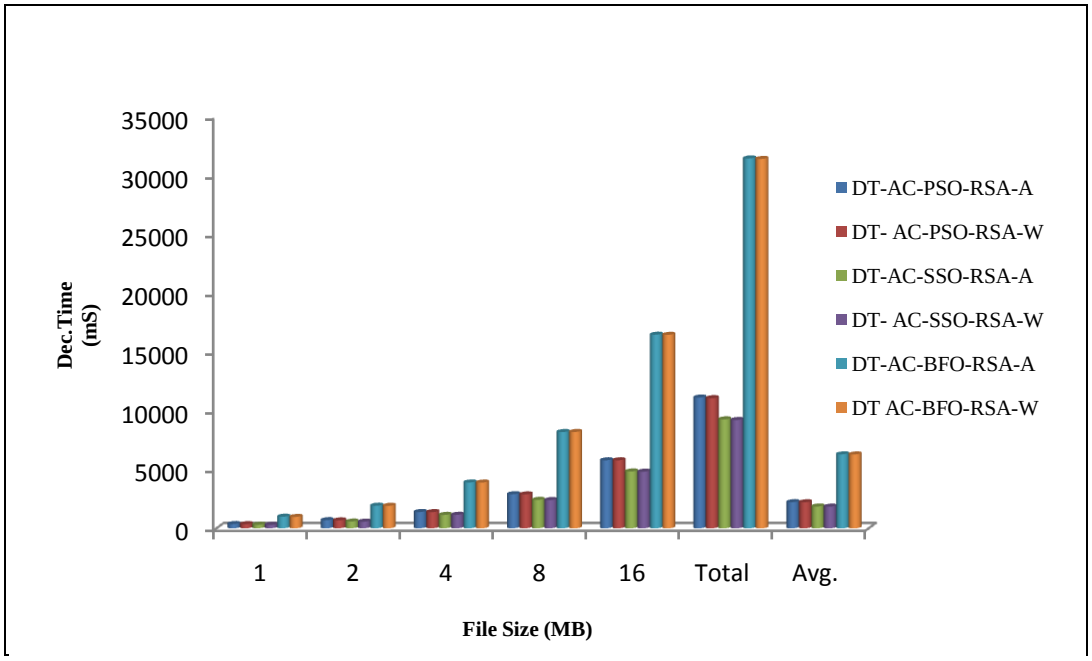


Fig. 7.3: Graph showing the Decryption time using AC-PSO-RSA, AC-SSO-RSA &AC-BFO-RSA with Android Vs Windows Emulator

- DT-AC-PSO-RSA-A is 1.005 times faster than DT- AC-PSO-RSA-W
- DT-AC-SSO-RSA-A is 1.006 times faster than DT- AC-SSO-RSA-W
- DT-AC-BFO-RSA-A is 1.001 times faster than DT- AC-BFO-RSA-W

It is also found that AC based on PSO, SSO and BFO improves the performance of DT when the said methods are used in RSA with A and W emulators. Further, DT-AC-SSO-RSA-W take less DT when they are compared with other proposed methods.

Table 7.4: Decryption time using AC-PSO-ECC, AC-SSO-ECC &AC-BFO-ECC with Android Vs Windows Emulator

FILE SIZE (MB)	DT-AC-PSO-ECC-A (mW)	DT -AC-PSO-ECC -W (mW)	DT -AC-SSO- ECC -A (mW)	DT-AC-SSO-ECC -W (mW)	DT -AC-BFO-ECC -A (mW)	DT - AC-BFO-ECC -W (mW)
1	543	525	422	422	1229	1218
2	1045	1017	800	792	2401	2381
4	2050	2055	1598	1596	4856	4854
8	4323	4318	3369	3361	10202	10201
16	8694	8681	6743	6734	20496	20476
Total	16655	16596	12932	12905	39184	39130
Avg.	3331	3319.2	2586.4	2581	7836.8	7826

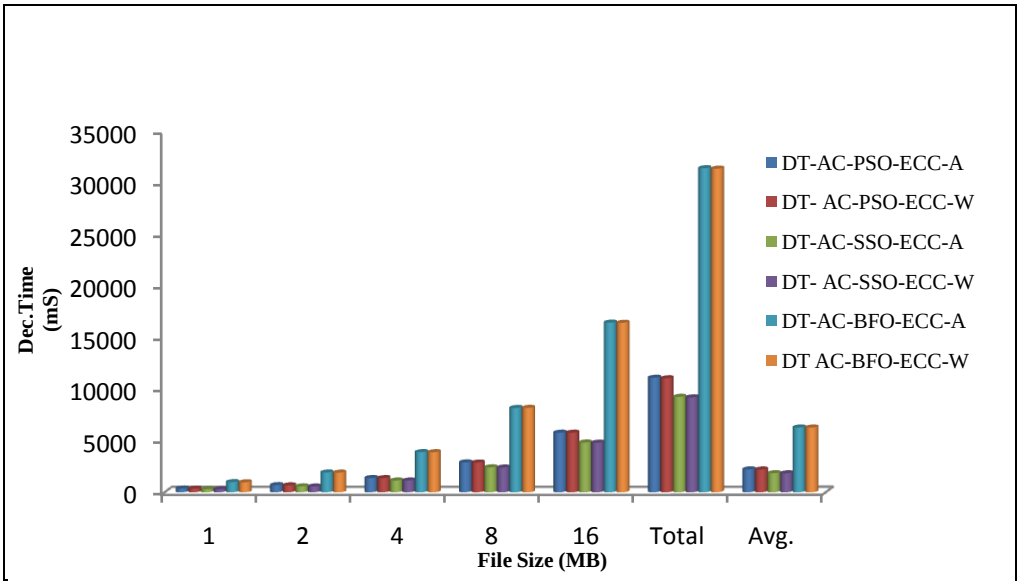


Fig. 7.4: Graph showing the Decryption time using AC-PSO-ECC, AC-SSO-ECC &AC-BFO-ECC with Android Vs Windows Emulator

- DT-AC-PSO-ECC-A is 1.004 times faster than DT- AC-PSO-ECC-W
- DT-AC-SSO-ECC-A is 1.002 times faster than DT- AC-SSO-ECC-W
- DT-AC-BFO-ECC-A is 1.001 times faster than DT- AC-BFO-ECC-W

It is also found that AC based on PSO, SSO and BFO improves the performance of DT when the said methods are used in ECC with A and W emulators. Further, DT-AC-SSO-RSA-W take less DT when they are compared with other proposed methods.

Table 7.5: Encryption power using AC-PSO-RSA, AC-SSO-RSA &AC-BFO-RSA with Android Vs Windows Emulator

FILE SIZE (MB)	EP-AC-PSO-RSA-A (mW)	EP-AC-PSO-RSA-W (mW)	EP-AC-SSO-RSA-A (mW)	EP- AC-SSO-RSA-W (mW)	EP-AC-BFO-RSA-A (mW)	EP- AC-BFO-RSA-W (mW)
1	1040	1023	865	850	343	336
2	2040	2012	1698	1692	649	666
4	4098	4094	3413	3412	1308	1307
8	8647	8631	7202	7175	2750	2747
16	17317	17308	14436	14422	5489	5489
Total	33142	33068	27614	27551	10539	10545
Avg.	6628	6613.6	5523	5510.2	2108	2109

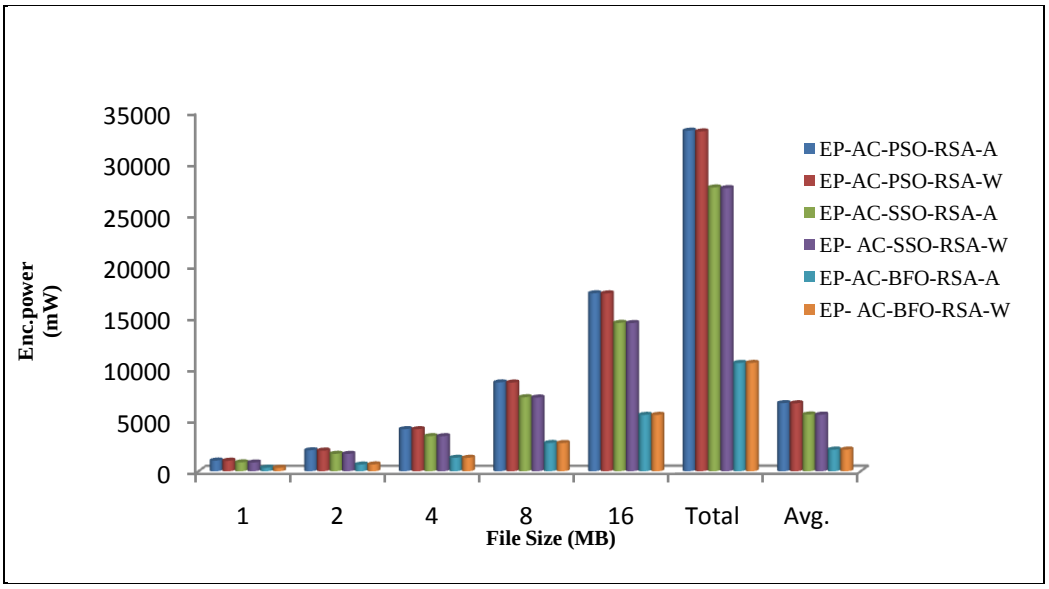


Fig. 7.5: Graph showing the Encryption power using AC-PSO-RSA, AC-SSO-RSA &AC-BFO-RSA with Android Vs Windows Emulator

- EP-AC-PSO-RSA-A is 1.002 times faster than EP-AC-PSO-RSA-W
- EP-AC-SSO-RSA-A is 1.002 times faster than EP- AC-SSO-RSA-W
- EP-AC-BFO-RSA-A is is almost same as EP- AC-BFO-RSA-W

It is also found that AC based on PSO, SSO and BFO improves the performance of EP when the said methods are used in RSA with A and W emulators. Further, EP-AC-BFO-RSA-A and EP-AC-BFO-RSA-W take less EP when they are compared with other proposed methods.

Table 7.6: Encryption power using AC-PSO-ECC, AC-SSO-ECC &AC-BFO-ECC with Android Vs Windows Emulator

FILE SIZE (MB)	EP-AC-PSO-ECC-A (mW)	EP-AC-PSO- ECC-W (mW)	EP-AC-SSO- ECC-A (mW)	EP- AC-SSO- ECC-W (mW)	EP-AC-BFO- ECC-A (mW)	EP- AC-BFO- ECC-W (mW)
1	1555	1533	1293	1300	395	401
2	3052	3015	2548	2527	782	777
4	6156	6150	5122	5122	1549	1561
8	12949	12925	10807	10762	3277	3277
16	26001	25952	21664	21643	6563	6544
Total	49713	49575	41434	41354	12566	12560
Avg.	9943	9915	8287	8270.8	2513	2512

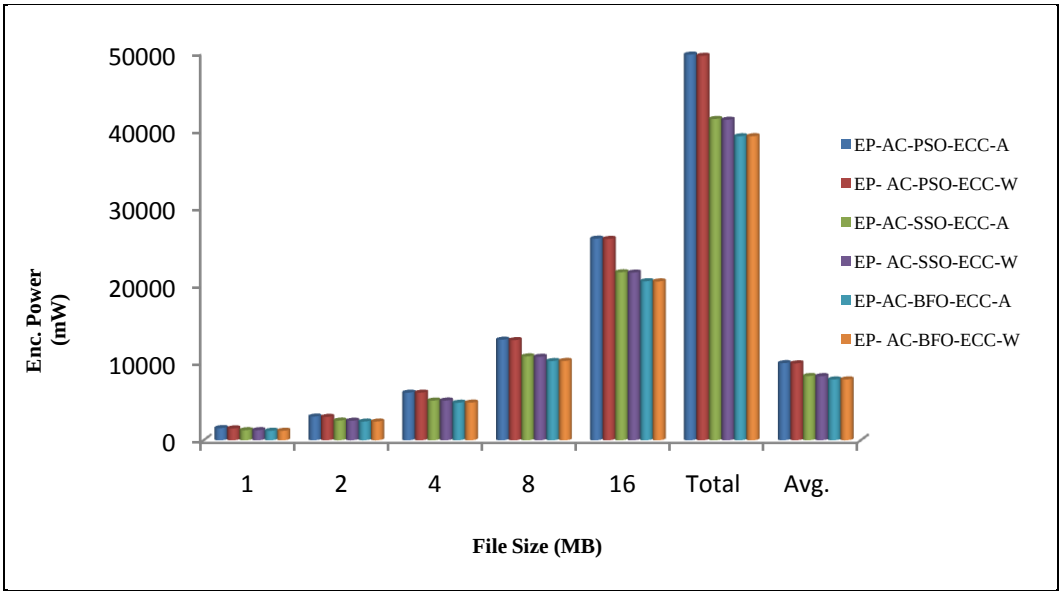


Fig. 7.6: Graph showing the Encryption power using AC-PSO-ECC, AC-SSO-ECC &AC-BFO-ECC with Android Vs Windows Emulator

- EP-AC-PSO-ECC-A is 1.002 times less than EP-AC-PSO- ECC-W
- EP-AC-SSO- ECC-A is 1.002 times less than EP- AC-SSO- ECC-W
- EP-AC-BFO- ECC-A is is almost same as EP- AC-BFO- ECC-W

It is also found that AC based on PSO, SSO and BFO improves the performance of EP when the said methods are used in ECC with A and W emulators. Further, EP-AC-BFO-ECC-A and EP-AC-BFO-ECC-W take less EP when they are compared with other proposed methods.

Table 7.7: Decryption power using AC-PSO-RSA, AC-SSO-RSA &AC-BFO-RSA with Android Vs Windows Emulator

FILE SIZE (MB)	DP-AC-PSO-RSA-A (mW)	DP-AC-PSO-RSA-W (mW)	DP-AC-SSO-RSA-A (mW)	DP- AC-SSO-RSA-W (mW)	DP-AC-BFO-RSA-A (mW)	DP- AC-BFO-RSA-W (mW)
1	363	354	288	293	334	322
2	696	676	577	570	654	637
4	1382	1375	1151	1150	1315	1305
8	2887	2894	2410	2394	2729	2740
16	5785	5786	4821	4812	5492	5495
Total	11113	11085	9247	9219	10524	10499
Avg.	2223	2217	1849	1843.8	2105	2099.8

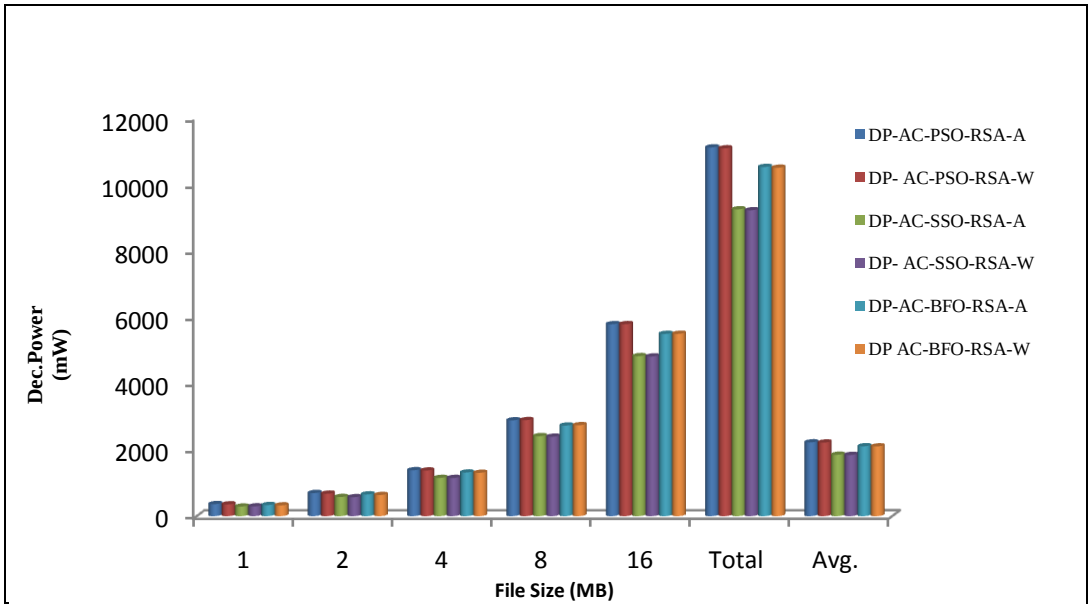


Fig. 7.7: Graph showing the Decryption power using AC-PSO-RSA, AC-SSO-RSA &AC-BFO-RSA with Android Vs Windows Emulator

- DP-AC-PSO-RSA-A is 1.003 times less than DP-AC-PSO-RSA-W
- DP-AC-SSO-RSA-A is 1.003 times less than DP- AC-SSO-RSA-W
- DP-AC-BFO-RSA-A is 1.002 times less than DP- AC-BFO-RSA-W

It is also found that AC based on PSO, SSO and BFO improves the performance of DP when the said methods are used in ECC with A and W emulators. Further, DP-AC-SSO-RSA-W take less DP when they are compared with other proposed methods.

Table 7.8: Decryption power using AC-PSO-ECC, AC-SSO-ECC &AC-BFO-ECC with Android Vs Windows Emulator

FILE SIZE (MB)	DP-AC-PSO-ECC-A (mW)	DP-AC-PSO-ECC-W (mW)	DP-AC-SSO- ECC-A (mW)	DP- AC-SSO- ECC-W (mW)	DP-AC-BFO- ECC-A (mW)	DP- AC-BFO- ECC-W (mW)
1	532	529	414	415	394	405
2	1024	1013	803	786	786	764
4	2066	2064	1598	1594	1559	1551
8	4332	4316	3375	3348	3264	3265
16	8677	8659	6745	6742	6553	6545
Total	16631	16581	12935	12885	12556	12530
Avg.	3326	3316.2	2587	2577	2511	2506

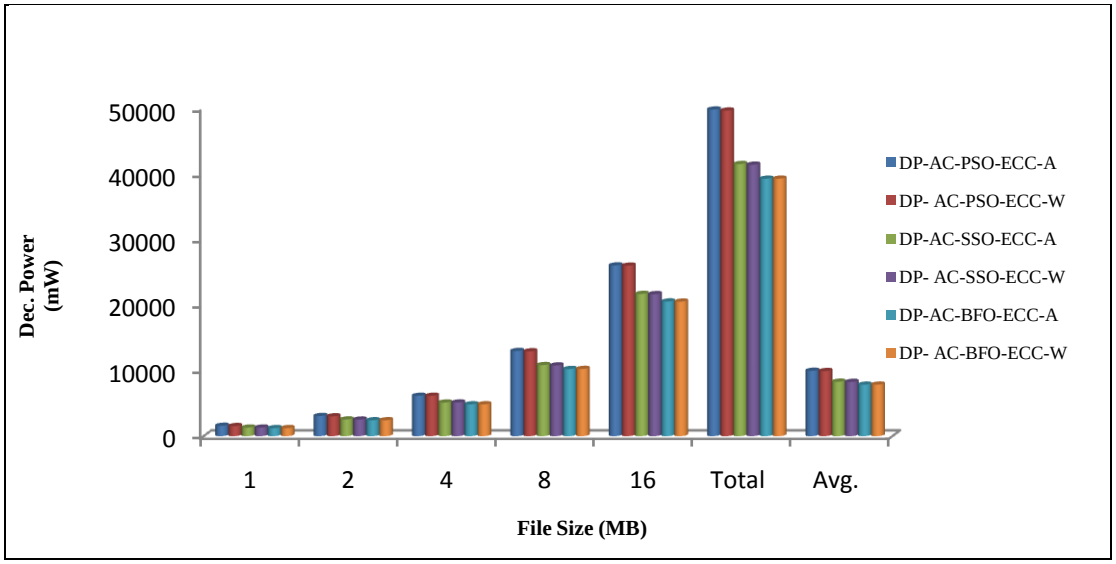


Fig. 7.8: Graph showing the Decryption power using AC-PSO-ECC, AC-SSO-ECC &AC-BFO-ECC with Android Vs Windows Emulator

- DP-AC-PSO-ECC-A is 1.003 times less than DP-AC-PSO- ECC-W
- DP-AC-SSO- ECC-A is 1.004 times less than DP- AC-SSO- ECC-W
- DP-AC-BFO- ECC-A is 1.002 times less than DP- AC-BFO- ECC-W

It is also found that AC based on PSO, SSO and BFO improves the performance of DP when the said methods are used in ECC with A and W emulators. Further, DP-AC-BFO-ECC-A and DP-AC-BFO-ECC-A take less DP when they are compared with other proposed methods.

Table 7.9: Security using AC-PSO-RSA, AC-SSO-RSA &AC-BFO-RSA with Android Vs Windows Emulator

FILE SIZE (MB)	SE-AC-PSO-RSA-A (%)	SE-AC-PSO-RSA-W (%)	SE-AC-SSO-RSA-A (%)	SE- AC-SSO-RSA-W (%)	SE-AC-BFO-RSA-A (%)	SE-AC-BFO-RSA-W (%)
1	95	93	93	93	94	95
2	92	92	92	91	92	91
4	90	91	91	90	91	90
8	89	89	90	89	90	89
16	89	88	89	88	88	89
Total	91	90.6	91	90.2	91	90.8
Avg.	95	93	93	93	94	95

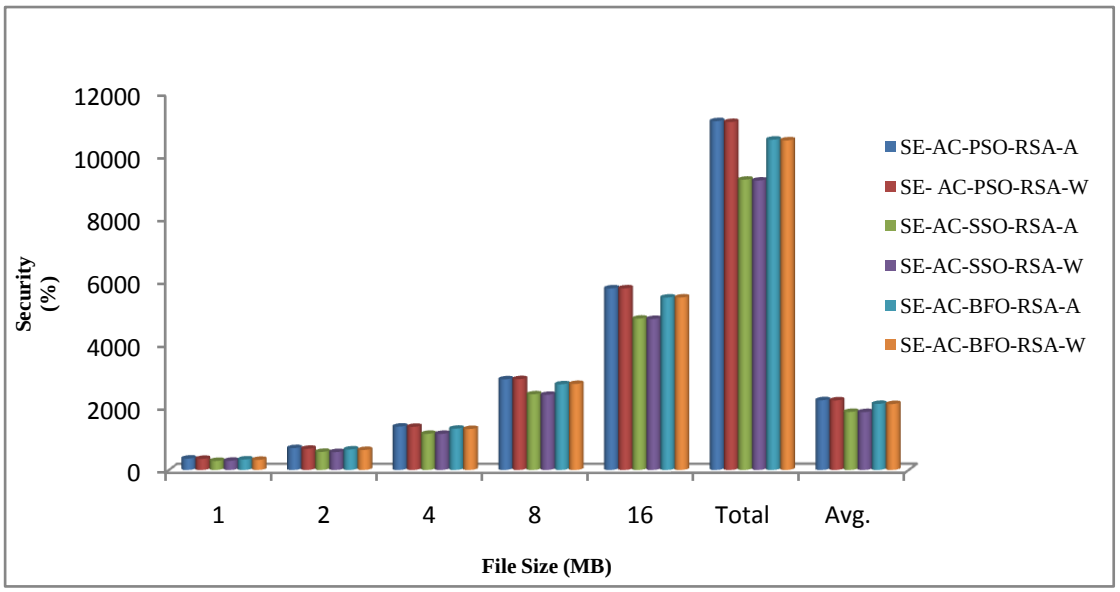


Fig. 7.9: Graph showing the Security using AC-PSO-RSA, AC-SSO-RSA &AC-BFO-RSA with Android Vs Windows Emulator

- SE-AC-PSO-RSA-A is 1.022 times more than SE-AC-PSO-RSA-W
- SE-AC-SSO-RSA-A is same as SE- AC-SSO-RSA-W
- SE-AC-BFO-RSA-W is 0.989 times more than SE-AC-BFO-RSA-A

It is also found that AC based on PSO, SSO and BFO provides more SE when the said methods are used in RSA with A and W emulators. SE-AC-PSO-RSA-A, SE-AC-BFO-RSA-A and SE-AC-BFO-RSA-W provides almost same level of security i.e., 94% when they are compared with other proposed methods.

Table 7.10: Security using AC-PSO-ECC, AC-SSO-ECC &AC-BFO-ECC with Android Vs Windows Emulator

FILE SIZE (MB)	SE-AC-PSO-ECC-A (%)	SE-AC-PSO-ECC-W (%)	SE-AC-SSO- ECC-A (%)	SE- AC-SSO- ECC-W (%)	SE-AC-BFO- ECC-A (%)	SE-AC-BFO- ECC-W (%)
1	95	95	93	97	96	97
2	94	93	92	93	93	93
4	92	93	91	92	93	93
8	91	91	90	92	91	91
16	91	91	89	91	91	91
Total	92.6	92.6	91	93	92.8	93
Avg.	95	95	93	97	96	97

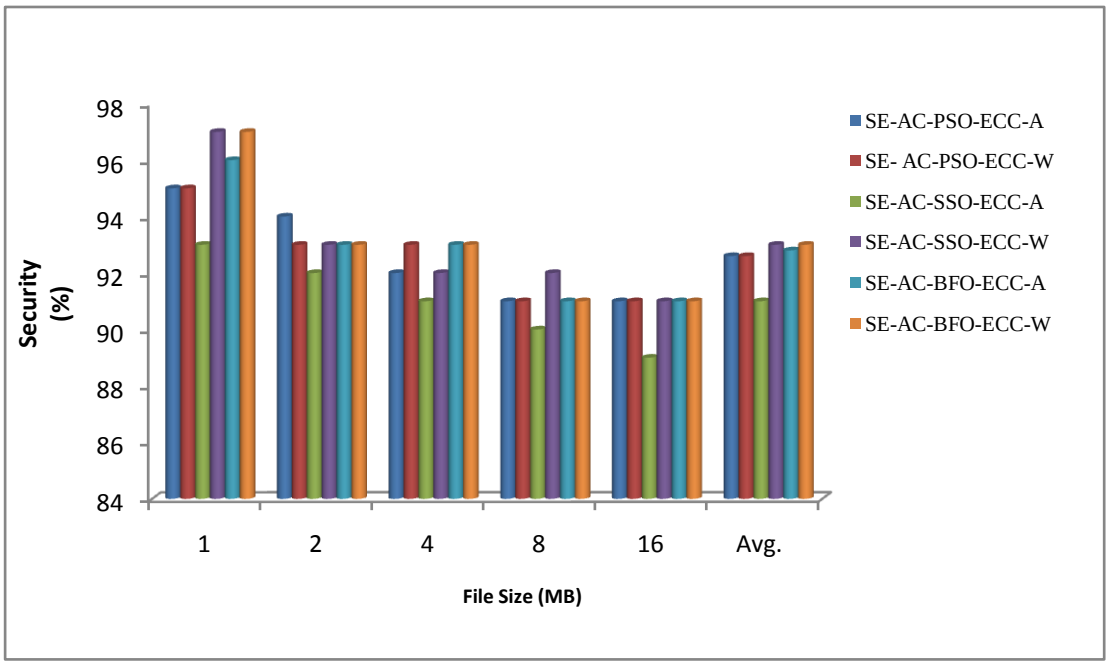


Fig. 7.10: Graph showing the Security using AC-PSO-ECC, AC-ECC-RSA &AC-BFO-ECC with Android Vs Windows Emulator

- SE-AC-PSO-ECC-A is same as SE-AC-PSO- ECC-W
- SE-AC-SSO- ECC-W is 1.021 times more than SE- AC-SSO- ECC-A
- SE-AC-BFO- ECC-W is 0.990 times more than SE-AC-BFO- ECC-A

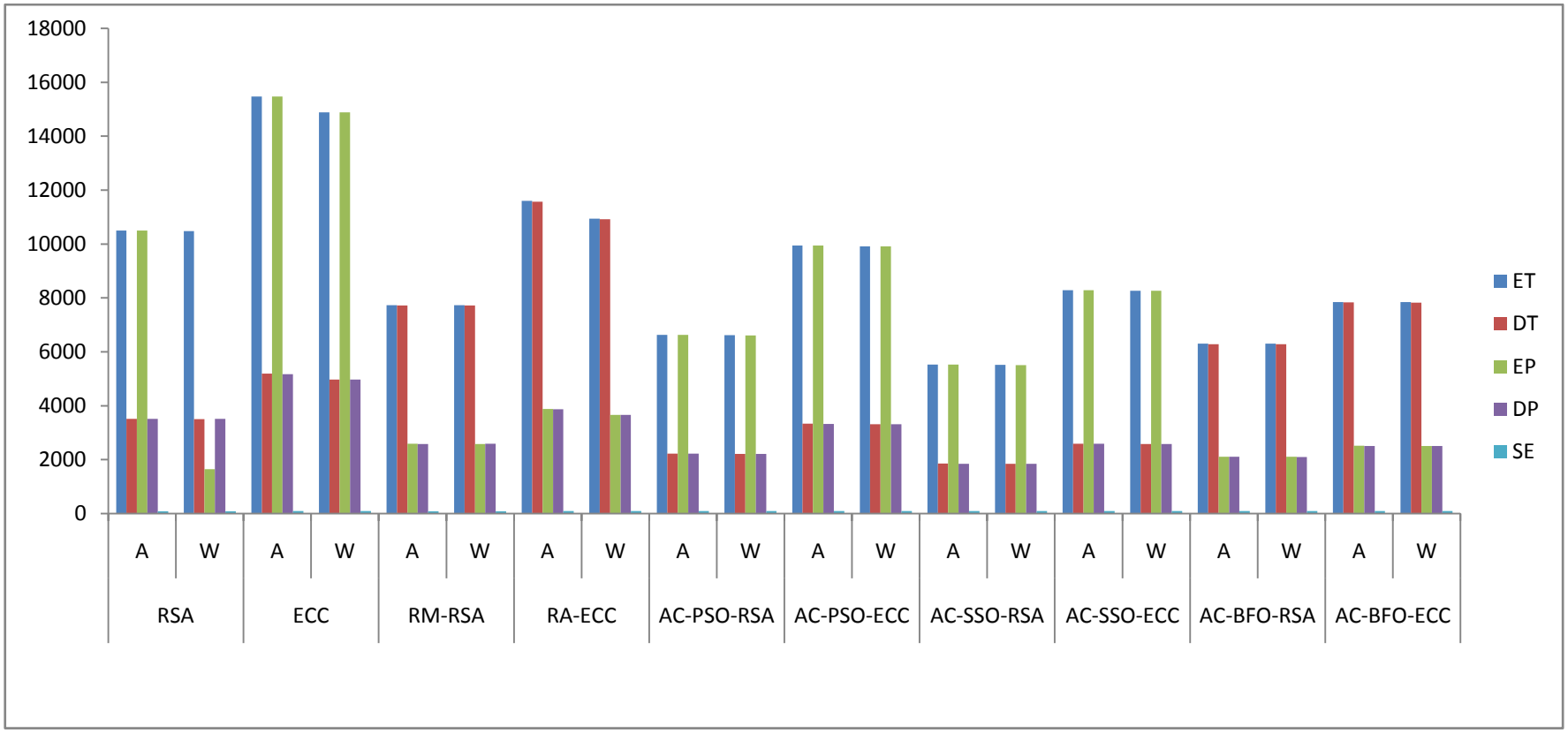
It is also found that AC based on PSO, SSO and BFO provides more SE when the said methods are used in ECC with A and W emulators. SE-AC-SSO-ECC-A and SE-AC-BFO-ECC-W provides almost same level of security i.e., 97% when they are compared with other proposed methods.

Table 7.11 shows the comparison of ET, DT, EP, DP and SE while using Android and Windows emulator respectively and their graphical representations are shown in fig.7.11.

From table 7.11, it is revealed that AC-SSO-RSA-W takes less ET and DT. It is also found that, AC-SSO-RSA-W consumes less power in both EP and DP. While considering SE, it is proved that AC-SSO-ECC-A and AC-BFO-ECC-W provides almost equal i.e., 93%.

Table 7.11: Android Vs Window OS Emulator

Operation	Existing Method				Proposed Methods											
	RSA		ECC		AC-PSO-RSA		AC-PSO-ECC		AC-SSO-RSA		AC-SSO-ECC		AC-BFO-RSA		AC-BFO-ECC	
	A	W	A	W	A	W	A	W	A	W	A	W	A	W	A	W
ET	10501.4	10475	15474	14885	6634	6614	9948.8	9917	5524.4	5513	8289	8268	6302.4	6301	7842.2	7843
DT	3510.6	3499	5194	4973.6	2222.6	2212.2	3331	3319.2	1852.2	1841.8	2586.4	2581	6288.4	6279.6	7836.8	7826
EP	10502	1643	15474	14885.8	6628	6613.6	9943	9915	5523	5510.2	8287	8270.8	2108	2109	2513	2512
DP	3511	3508.8	5174	4975.8	2223	2217	3326	3316.2	1849	1843.8	2587	2577	2105	2099.8	2511	2506
SE	88.6	88.8	90.6	90.8	91	90.6	92.6	92.6	91	90.2	92.8	93	91	90.8	92.8	93



ET : Encryption Time DT : Decryption Time EP : Encryption Power DP : Decryption Power SE: Security

Fig. 7.11: Graph Showing the overall performance of Android Vs Window OS Emulator

CHAPTER – VIII

CONCLUSION

Mobile computing is a technology which can defined as at anywhere, any time and any place anybody can access it. In future, all activities can be controlled through these devices, which allow the user without any physical connection to complete the tasks. These devices will enable the transmission of voice, video and data between human and the computer. It always helps to stay connected to the world with a wide range of users through the internet. Even though, all sensitive informations are transmitted through these handheld mobile devices but they have limited battery power, storage and some security threats occur. Hence, the five parameters viz., ET, DT, EP, DP and SE have been taken in this research. If the said processes take more time, it will degrade the performance and will eventually decrease the life of mobile devices. It is noted that there is a directly proportional relationship between operational time and power consumption.

Thus, it is essential to minimize the operational time. One way of achieving is to use the optimal AC. To generate the optimal AC for the integer (keys in cryptographic algorithms), the concepts used in BIAs viz., PSO, SSO and BFO are taken and they are incorporated into the public-key algorithms RSA and ECC with two different mobile emulators Android and Windows. The proposed algorithms are termed as AC-PSO-RSA, AC-PSO-ECC, AC-SSO-RSA, AC-SSO-ECC, AC-BFO-RSA and AC-BFO-ECC. They are tested for said parameters.

8.1 Summary of the Contributions

The contributions made in this research work are summarised as follows:

i) In the first proposed AC-PSO method, each particles represents the AC, and the fitness function represents the length of AC. Even though, too may ACs are generated for the encryption and decryption key of RSA and k[P] of ECC, only optimal length ACs are alone is considered in this method. The generated ACs are incorporated into RSA and ECC with two different emulators. Experimental result clearly indicate that (i) ET-AC-PSO-RSA-A takes less ET, and DT-AC-PSO-RSA-A and DT-AC-PSO-RSA-W take less DT than the existing and the proposed methods. (ii) when EP is concerned, EP-AC-PSO-RSA-W and DP-AC-PSO-RSA-A take less EP and DP than the existing and the proposed methods and (iii) regarding the security parameter, SE-AC-PSO-ECC-A and SE-AC-PSO-ECC-W provide almost same but more security than the existing and the proposed methods.

ii) In the second proposed AC-SSO method, CPs are taken as ACs and again the length of the AC is taken as fitness function. Only optimal length ACs generated by this method are incorporated into RSA and ECC. From the experimental results, it is observed that (i) ET-AC-SSO-RSA-W takes less time for both encryption and decryption than the existing and the proposed methods. (ii) when EP is concerned, EP-AC-SSO-RSA-A takes less EP and DP-AC-SSO-RSA-A and DP-AC-SSO-RSA-W take less DP than the existing and the proposed methods and (iii) regarding the security parameter, SE-AC-SSO-ECC-W provides more security than the existing and the proposed methods.

iii) AC-BFO is the third proposed method in which each bacterium is considered as AC and the optimal ACs are produced using the processes viz., chemotaxis, reproduction, and elimination-dispersal. As the ultimate aim of this proposed method is to reduce the operational speed, energy consumption and enchaining the security, this method is also used for the same to avoid the customer's impatience and dissatisfaction. The experimental results clearly show that (i) ET-AC-BFO-RSA-W takes less ET and DT than the existing and the proposed methods (ii) when EP is concerned, EP-AC-BFO-RSA-W and DP-AC-BFO-RSA-A take less EP and DP-AC-BFO-RSA-A and DP-AC-BFO-RSA-W take the same and less DP than the existing and the proposed methods and (iii) regarding the security parameter, SE-AC-BFO-ECC-A and SE-AC-PSO-ECC-W take almost same but more security than the existing and the proposed methods.

From the three proposed methods, it is concluded that AC-SSO-RSA-W is for ET, DT, EP and DP are better than others with respect to time. While considering SE, it is proved that AC-SSO-ECC-A and AC-BFO-ECC-W provides almost same security level i.e., 93% than others.

8.2 Future Research Directions

This work can be extended in mobile cloud computing due to the serious limitations of memory space, battery power for energy consumption as well as the resource optimization without compromising the security in mobile devices.

8.3 End Note

Mobile computing technology can reach at any part of the world to attain its destiny. Users can feel very comfortable from any location as they are connected to a secure

network. This technology acts as a major part of Information Communication and Technology (ICT). Mobile functionality available today but their performances need to be safe and secured means this can lead to attain its heights. This research work has built a new three proposed BIAs based AC methods to enhance the operational speed while using any cryptographic algorithms. No cryptographic algorithms have proved to use such methods to enhance the operational speed in any literature. These ideas used are unique, novel, innovative and original. This work is nonexistent in any literature and the same is endorsed by a few journals and conferences for its veracity.

REFERENCES

[1]. J Jang-accard and Surya Nepal, "A Survey of emerging threats in Cybersecurity", *Journal of Computer and System Sciences*, 2014, 80, pp.973-993.

[2]. International Telecommunication Union, CCITT - The International Telegraph and Telephone Consultative Committee - Security Architecture for Open Systems Interconnection for CCITT Applications", *Recommendation X.800*, Geneva, 1991.

[3]. Gurkan Gur et. al., "Security Analysis of Computer Nerworks: Key concepts and methodologies", *Modeling and Simulation of Computer Networks and Systems - Methodologies and Applications*, 2015, pp.861-898.

[4]. William Stallings, "*Cryptography and Network Security*", 2005, 4th edition, pp.209.

[5]. Rob Stubbs, "Classification of Cryptographic Keys", *A Framework for Designing Cryptographic Key Management Systems*, 2018.

[6]. Massoud Sokouti et al., "An approach in improving transposition cipher system", *Indian Journal of Science and Technology*, 2009.

[7]. Hans Delf and Helmut Knel, "*Introduction to Cryptography - Principles and Applications*, Second Edition, Springer, 2007.

[8]. Neal Koblitz, "*A Course in Number Theory and Cryptography*", Springer Verlag, 1994.

[9]. Gary C. Kessler, "An Overview of Cryptography", *Handbook on Local Area Networks*, 1998

[10]. Britannica, The Editors of Encyclopaedia. "cipher". *Encyclopedia Britannica*, 14 Jun. 2021.

[11]. Limor Elbaz, "Using Public Key Cryptography in Mobile Phones", *White Paper,Discretix Technologies Ltd., Advanced security solutions for constrained environments*, October 2002.

[12]. Miller V, "Use of elliptic curves in cryptography", *Advances in Cryptology - CRYPTO '85*, Lecture Notes in Computer Science, 1986.

[13]. Evan Dummit, "*Cryptography (part 3): Discrete Logarithms in Cryptography 2016*", Vol. 1.01, pp. 1-13.

[14]. Jayaprakash Kar & Banshidhar Majhi, “An Efficient Password Security of Multi-Party key exchange protocol based on ECDLP”, *International Journal of Computer Science and Security (IJCSS)*, Vol.1, Issue 5, Sep. 2009.

[15]. Immons, Gustavus J, "RSA encryption", *Encyclopedia Britannica*, 3 Aug. 2012.

[16]. Koblitz, N, "Ellipti Curve Cryptosystems", *Mathematics of Computation*, 48 (177): 203-209, doi:10.2307/2007884. JSTOR 2007884.

[17]. Bruno P.S. Rocha et al., “Adaptive Security protocol selection for mobile computing”, *Journal of Network and Computer Applications*, 33, 2010, pp. 569.

[18]. Bezboruah, Tulshi, "Mobile Computing", *The Emerging Technology, Sensing, Challenges and Applications*, 2011, Vol. 4, pp. 165-174.

[19]. David Kleidermacher, Mike Kleidermacher, "Embedded Cryptograph", *Embedded Systems Securit*y, 2012.

[20]. Jyotsna Dei, Anindya Sen, "Investigation on Trends of Mobile Operating Systems", *International Journal of Engineering Research & Technology (IJERT)*, Vol.4 Issue 07, July 2015.

[21]. Vijay K. Garg, "*Wireless Communication & Networking*", 2007.

[22]. Mooseeop Kim et al., “Design of Cryptographic Hardware Architecture for Mobile Computing”, *Journal of Information Processing Systems*, vol. 5, no. 4, Dec. 2009.

[23]. Maurice Mignotte, "A Note on Addition Chains", *International Journal of Algebra*, Issue 5(6), 2011.

[24]. Neil Michael Clift, "Calculating Optimal Addition Chains”, *Journal of Computing*, Springer, 91, 2011, pp. 265–284.

[25]. K Mani, M Viswambari, "A New Method of Generating Optimal Addition Chain Based on Graph, *International Journal of Mathematical Sciences and Computing*", Vol. 2, 2017, pp. 37-54.

[26]. P Suradhakameswari and B Ravitheja, Addition Chain for Lucas sequences with Fast Computation Method, *International Journal of Applied Engineering Research*, Issue 13(11), 2018, pp. 9413–9419.

[27]. Dustin Moody and Amadou Tall, On Addition-Subtraction Chains of Numbers With Low Hamming Weight*”, Number Theory Mathematics*, Vol. 25, 2019, pp. 155-168.

[28]. Floreano and Mattiussi, "*Bio-inspired artificial intelligence: Theories, methods, and technologies*", 2008.

[29]. Xian-Bing Meng, X.Z. Gao, Lihua Lu, Yu Liu, Hengzhen Zhang. "A new bio-inspired optimisation algorithm: Bird Swarm Algorithm", *Journal of Experimental & Theoretical Artificial Intelligence*, 2016.

[30]. Ke-Lin Du and M. N. S. Swamy. "Particle Swarm Optimization", *Search and Optimization by Metaheuristics*, Springer, 2016.

[31]. Wei-Chang Yeh, Wei-Ting Lin, Chyh-Ming Lai, Yen-Chin Lee, Yuk Ying Chung, Jsen-Shung Lin. "Application of simplified swarm optimization algorithm in deteriorate supply chain network problem", *Evolutionary Computation (CEC) - IEEE,* 2016.

[32]. Swagatam Das, ArijitBiswas, SambartaDasgupta, and Ajith Abraham, "Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications", *Foundations of Computational Intelligence, Springerlink.com, Springer-Verlag Berlin Heidelberg*, 2009, pp. 23–55.

[33]. Kevin M. Paasino, "Bacterial Foraging Optimization", *International Journal of Swarm Intelligence Research*, 1(1), 2010, pp.1-16.

[34]. Mavridis I., Pangalos G., "Security Issues in Mobile computing Paradigm". 1997, http://www.researchgate.net.

[35]. Erik Olson and Woojin Yu, "Encryption for Mobile computing", 2000.

[36]. Wendy Chou, "Elliptic Curve Cryptography and its applications to Mobile Devices", 2000.

[37]. Limor Elbaz, "Using Public Key Cryptography in Mobile Phones", *White Paper,Discretix Technologies Ltd., Advanced security solutions for constrained environments*, October 2002.

[38]. Dharma P. Agrawal et al., "Secure Mobile Computing", S.R. Das, S.K. Das (Eds.): IWDC 2003, *Springer-Verlag., LNCS 2918*, 2003, pp.265- 278.

[39]. WHanping Lufei and Weisong Shi, "An Adaptive Encryption Protocol in Mobile Computing", *Wireless/Mobile Network Security, Springer*, 2006.

[40]. Abhishek Kumar Gupta, "Challenges of Mobile computing", *Proceedings of 2nd National Conference on Challenges & Opportunities in Information Technology RIMT – IET*, Mandi Gobindgarth, March 29, 2008.

[41]. S. Krishna Mohan Rao and Dr. A Venugopal Reddy, "Data Dissemination in Mobile Computing Environment", *BIJIT*, Bharati Vidyapeeth's Institute of Computer applications and Management (BVICAM), New Delhi, Vol. 1, No. 1, January 2009.

[42]. M. Razvi Doomun, and KMS Soyjaudah, "Analytical Comparison of Cryptographic Techniques for Resource-Constrained Wireless Security", *International Journal of Network Security*, Vol.9, No.1, July 2009, pp. 82–94.

[43]. Jayaprakash Kar & Banshidhar Majhi, "An Efficient Password Security of Multi-Party key exchange protocol based on ECDLP", *International Journal of Computer Science and Security (IJCSS)*, Vol.1, Issue 5, Sep. 2009.

[44]. Mooseeop Kim et al., "Design of Cryptographic Hardware Architecture for Mobile Computing", *Journal of Information Processing Systems*, Vol. 5, No. 4, Dec. 2009.

[45]. Bruno P.S. Rocha et. al., "Adaptive Security protocol selection for mobile computing", *Journal of Network and Computer Applications,* 2010, pp. 569.

[46]. Sathish Alampalayam Kumar, "Classification and Review of Security Schemes in Mobile Computing", *Wireless Sensor Network*, June 2010, pp.419-440.

[47]. Sameer Hasan Al-Bakri, Gazi Mahabubul Alam et al., "Securing peer-to-peer mobile communications using public key cryptography: New security strategy", *International Journal of the Physical Sciences*, Vol. 6(4), Feb. 2011, pp. 930-938.

[48]. Rahat Afreen and S.C. Mehrotra, "A Review on Elliptic Curve Cryptography for Embedded Systems", *International Journal of Computer Science & Information Technology*, Vol. 3, No 3, June 2011.

[49]. Helena Rifa-Pous and Jordi Herrera-Joancomarti, "Computational and Energy Costs of Cryptographic Algorithms on Handheld Devices", *Future Internet*, 2011, Vol.3, pp.31-48.

[50]. Jagdish Bhatta and Lok Prakash Pandey, "Performance Evaluation of RSA Variants and Elliptic Curve Cryptography on Handheld Devices", *International Journal of Computer Science and Network Security*, Vol. 11, No. 11, Nov. 2011.

[51]. K. Sathish Kumar et. al., “An Experimental Study on Energy Consumption of Cryptographic Algorithms for Mobile Hand-Held Devices”, *International Journal of Computer Applications*, Vol. 40, No.1, Feb. 2012.

[52]. Masoud Nosrati et. al., “Mobile and Operating Systems”, *Computing: Principles, Devices World Applied Programming*, Vol. 2, Issue 7, July 2012.

[53]. Ravinder Singh Mann et al., “A Comparative Evaluation of Cryptographic Algorithms”, *Int. J. Computer Technology & Applications*, Vol. 3(5), Oct. 2012, pp. 1653-1657.

[54]. Giripunje et al., “Comprehensive Security System for Mobile Network Using Elliptic Curve Cryptography over GF (p)", *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 3, Issue 5, May 2013, pp. 704-713.

[55]. Ameya Nayak, “Android Mobile Platform Security and Malware Survey”, *IJRET: International Journal of Research in Engineering and Technology*, Vol. 02 Issue 11, Nov. 2013.

[56]. Srikanth Pullela, “Security Issues in Mobile computing”, *International Journal of Research in Engineering and Technology*, Vol. 02, Issue: 11, Nov. 2013.

[57]. V. Gayoaso Martinez and L. Hernandez Encinas, “Implementing ECC with Java Standard Edition 7”, *International Journal of Computer Science and Artificial Intelligence*, Dec. 2013, Vol. 3, Issue. 4, pp. 134-142.

[58]. Muhammad Waseem Khan, “SMS Security in Mobile Devices: A Survey”, *Int. J. Advanced Networking and Applications*, Vol. 05, Issue 2, 2013, pp. 1873 -1882.

[59]. Ram Ratan Ahirwal and Manoj Ahke, “Elliptic Curve Diffie-Hellman Key Exchange Algorithm for Securing Hypertext Information on Wide Area Network”, *International Journal of Computer Science and Information Technologies*, Vol. 4(2), 2013, pp.363 – 368.

[60]. Sathish Kumar et. al., “An Asymmetric Authentication Protocol for Mobile Hand held Devices using ECC over Point Multiplication Method”, *International Journal of Advanced Research in Computer Science & Technology*, Vol. 2, Jan.–March 2014.

[61]. Hamed Khiabani et. al., "A Review on privacy, Security and Trust issues in Mobile Computing", *Collaborative outcome of University of Malaysia and MIMOS Berhad – Information Security Cluster*, 2014.

[62]. Seema P. Nakhate and R.M. Goudar, "Secure Authentication Protocol", *International Journal of Computer Networks and Communications Security*, Vol. 2, No. 4, April 2014, pp. 142 – 145.

[63]. Vishnu V and Shobha R, "Dynamic Cluster Head (CH) Node Election and Secure Data Transaction in CWSNs", *International Journal of Engineering Research*, Vol. 4, Issue Special 4, May 2015.

[64]. Tanmoy Kumar Bishoi et. al., "An Algorithm on Text Based Security in Modern Cryptography", *Journal of Computer Networking, Wireless and Mobile Communications (JCNWMC)*, Vol. 5, Issue 1, Jun 2015, pp. 9-14.

[65]. Sujithra M et. al., "Mobile Data Security: A Cryptographic Approach by Outsourcing Mobile data to Cloud", *Procedia Computer Science*, 2015, pp. 480-485.

[66]. Said Bouchkaren and Saiida Lazaar, "A New Iterative Secret Key Cryptosystem Based on Reversible and Irreversible Cellular Automata", *International Journal of Network Security*, Vol. 18, No. 2, Mar 2016, pp. 345-353.

[67]. Arbit and Ashwini Kumar, "Optimized Elliptic Curve Cryptography as Fine Balance for Wireless Sensor Network", *International Journal of Modeling and Optimization*, Vol.1, No. 4, October 2011.

[68]. Ahmed Tariq Sadiq, "Mutation-Based Particle Swarm Optimization (MPSO) to Attack Classical Cryptography Methods", *Journal of Computer Science and Technology Research*, Issue 2, March 2012., pp. 50-65.

[69]. Ahemed A, A. Esmin and Germano Lambert-Torres, "Application of Particle Swarm Optimization to optimal power systems", *International Journal of Innovative Computing, Information and Control*, Vol. 8, No.3(A), March 2012, pp. 1705-1716.

[70]. G.Prakash and Dr. M. Kannan, "Enhancing Security in Cryptographic in Smart Cards through Elliptic Curve Cryptography and Optimized Modified Matrix Encoding Algorithms", *Journal of Theoretical and Applied Information Technology*, Vol. 58, No.3, December 2013.

[71]. Cuevas, E., Cienfuegos, M., Zaldívar, D., Pérez-Cisneros, M. A swarm optimization algorithm inspired in the behaviour of the social-spider, *Expert Systems with Applications*, Vol. 40, No. 16, 2013.

[72]. Wilayat Khan, Habib Ullah and Riaz Hussain, "Energy Efficient Mutual Authentication Protocol for Handheld devices based on Public Key Cryptography", *International Journal of Computer Theory and Engineering*, Vol. 5, No. 5, October 2013.

[73]. Rangit j. Bhosale et al, "A Survey on Intrusion detection System for Mobile Ad-hoc Networks", *(IJCSIT) International Journal of Computer Science and Information Technologies,* Vol. 5, No. 6, 2014.

[74]. Swapna B. Sasi and N. Sivanandam, "A Survey on Cryptography using Optimization algorithms in WSNs", *Indian Journal of Science and Technology*, Vol. 8. No. 3, February 2015.

[75]. Dolly U. Jeswani and Swati G. Kale, "The Particle Swarm Optimization Based Linear Cryptanalysis of Advanced Encryption Standard Algorithm", *International Journal on Recent and Innovation Trends in Computing and Communication,* Vol. 3, April 2015.

[76]. Chia-Ling Huang and Wei-Chang Yeh," Simplified Swarm Optimization Algorithm for reliability redundancy allocation problems", *IEEE Computer Society*, 2015.

[77]. Ji Weidong and Zhu Songyu, "A Filtering Mechanism Based Optimization for Particle Swarm", *International Journal of u- and e-Service, Science and Technology*, Vol.9, No. 1, 2016.

[78]. Jin Yang, Fagui Liu, Jianneng Cao and Liangming Wang, "Discrete Particle Swarm Optimization Routing Protocol for Wireless Sensor Networks with Multiple Mobile Sinks", *MDPI journals , Sensors* 2016.

[79]. Joppe W. Bos, J. Alex Halderman, Nadia Heninger, Jonathan Moore, Michael Naehrig, Eric Wustrow. "Elliptic Curve Cryptography in Practice", *International Conference on Financial Cryptography and Data Security, Springer,* 2014.

[80]. Lijuan Li and Shuguo Li, "Fast inversion in GF(2m) with polynomial basis using optimal addition chains*", Circuits and Systems (ISCAS), IEEE*, 2017.

[81]. Joost Renes, Craig Costello and Lejla Batina, "Complete Addition Formulas for Prime Order Elliptic Curves", *Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer*, 2016.

[82]. Ke-Lin Du and M. N. S. Swamy. "Particle Swarm Optimization", *Search and Optimization by Metaheuristics, Springer*, 2016.

[83]. Floreano and Mattiussi, Bio-inspired artificial intelligence: Theories, methods, and technologies, 2008.

[84]. Micael Couceiro, Pedram Ghamisi. "Particle Swarm Optimization", *Fractional Order Darwinian Particle Swarm Optimization - Springer*, 2015.

[85]. Wei-Chang Yeh, Cyuan-Yu Luo, Chyh-Ming Lai, Chi-Ting Hsu, Yuk Ying Chung, Jsen-Shung Lin., "Simplified swarm optimization with modular search for the general multi-level redundancy allocation problem in series-parallel systems", *Evolutionary Computation (CEC) - IEEE*, 2016.

[86]. Fausto Meneses, Walter Fuertes, José Sancho, Santiago Salvador, Daniela Flores, Hernán Aules, Fidel Castro, Jenny Torres, Alba Miranda, Danilo Nuela. "RSA Encryption Algorithm Optimization to Improve Performance and Security Level of Network Messages", *International Journal of Computer Science and Network Security - IJCSNS*, 2016.

[87]. Nigel P. Smart. "Elliptic Curves", Cryptography Made Simple - *Springer*, 2015.

[88]. Nicholas M. Katz and Barry Mazur, "Arithmetic Moduli of Elliptic Curves", *Annals of Mathematic Studies, Princeton University Press*, 2016.

[89]. Turner Paul1, Thornton Steve. "Addition chains: A reSolve lesson", *Australian Senior Mathematics Journal*, 2017.

[90]. Stjepan Picek, Carlos A. Coello Coello, Domagoj Jakobovic, Nele Mentens. "Evolutionary Algorithms for Finding Short Addition Chains: Going the Distance", *Evolutionary Computation in Combinatorial Optimization - Springer*, 2016.

[91]. Stjepan Picek, Carlos A. Coello Coello, Domagoj Jakobovic, Nele Mentens. "Finding short and implementation-friendly addition chains with evolutionary algorithms", *Journal of Heuristics, Springer*, 2017.

[92]. Brian Koziel, Reza Azarderakhsh, David Jao, Mehran Mozaffari-Kermani. "On Fast Calculation of Addition Chains for Isogeny-Based Cryptography", *Information Security and Cryptology*, 2016.

[93]. K.Mani, M.Viswambari."A New Method of Generating Optimal Addition Chain Based on Graph", *I.J. Mathematical Sciences and Computing - MECS*, 2017.

[94]. Michal Pluhacek, Jakub Janostik, Roman Senkerik, Ivan Zelinka, Donald Davendra. "PSO as Complex Network—Capturing the Inner Dynamics—Initial Study", *Proceedings of the Second International Afro-European Conference for Industrial Advancement , Springer*, 2016.

[95]. Xian-Bing Meng, X.Z. Gao, Lihua Lu, Yu Liu, Hengzhen Zhang. "A new bio-inspired optimisation algorithm: Bird Swarm Algorithm", *Journal of Experimental & Theoretical Artificial Intelligence*, 2016.

[96]. Yanmin Liu, Chengqi Li, Xiangbiao Wu, Qingyu Zeng, Rui Liu, Tao Huang. "Particle Swarm Optimizer with Full Information", *Intelligent Computing Theories and Application, Springer*, 2016.

[97]. Wei-Chang Yeh, Wei-Ting Lin, Chyh-Ming Lai, Yen-Chin Lee, Yuk Ying Chung, Jsen-Shung Lin. "Application of simplified swarm optimization algorithm in deteriorate supply chain network problem", *Evolutionary Computation (CEC) - IEEE*, 2016.

[98]. Shreenath Acharya, Asha Shenoy, Macwin Lewis, Namrata Desai. "Analysis and Prediction of Application Usage in Android Phones", *Advances in Electrical, Electronics, Information, Communication and Bio-Informatics*", *IEEE*, 2016.

[99]. https://www.microsoft.com/en-us/download/details.aspx?id=53424**.**

[100]. Neal Koblitz, "A Course in Number Theory and Cryptography", *Springer Verlag*, 1994.

[101]. Koblitz N, "Elliptic curve cryptosystems", 1987.

[102]. Miller V, "Use of elliptic curves in cryptography", *Advances in Cryptology - CRYPTO '85, Lecture Notes in Computer Science*, 1986.

[103]. N Koblitz, Elliptic Curve Cryptosystems, Mathematics of Computation, 48, 1982, pp. 203-209.

[104]. I Blake, G Seroussi and NP Smart, Elliptic Curves in Cryptography, Ser. London Math. Soc. Lecture Note Series, Cambridge Univ. Press, 1999.

[105]. Hugo Volger, "Some Results on Addition/Subtraction Chains", *Information Processing Letter, Elsevier,* 1985.

[106]. Y H TsaiandY H Chin,"A Study of Some Addition Chain Problems", *International Journal of Computer Mathematics,* 22(02), 1987, pp. 117-134.

[107]. F Bergeron, J Berstel, S Brlek, and C Duboc, "Addition Chains Using Continued Fractions", *Journal of Algorithms, Elsevier,* 1989, pp. 403-412.

[108]. F Bergeron J Berstel and S Brlek, "Efficient Computation of Addition Chains", *Joumal de Theorie des Nombresde Bordeaux,* 6(1), 1994, pp. 21-38.

[109]. Donald E Knuth, "The Art of Computer Programming, Seminumerical Algorithms", 2(3), *Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA,* 1997.

[110]. Gordon DM, "A Survey of Fast Exponentiation Methods", *Journal of Algorithms,* 1998.

[111]. H Zantema, "Minimizing Sums of Addition Chains", *Journal of Algorithms, Elsevier,* 12(2), 1999, pp. 21-38.

[112]. Noboru Kunihiro and Hirosuke Yamamoto, "New Methods for Generation of Short Addition Chains" , *IEICE Transactions Fundamental,* 83(1), 2000.

[113]. Nareli Cruz-Cortés, Francisco Rodriguez-Henriquez, RaúlJuárez-Morales and Carlos A Coello- Coello, "Finding Optimal Addition Chains Using a Genetic Algorithm Approach", *Springer- Verlag*, 2005, pp. 208-215.

[114]. Cortés, Nareli & Trejo-Pérez, Daniel & Coello, Carlos. Handling Constraints in Global Optimization Using an Artificial Immune System, *Lecture Notes in Computer Science,* 2005, pp. 234-247.

[115]. Raveen R Goundar, Ken-ichiShiota, M Toyonaga, "New Strategy for Doubling - Free Short Addition-Subtraction Chain", *Mathematics,* 2008.

[116]. AlejandroLe'on-Javier, NareliCruz-Cort'es, Moreno-Armend ariz, and Sandra Orantes - Jim'enesz, Finding Minimal Addition Chains with a Particle Swarm Optimization Algorithm, *Advances in Artificial Intelligence, Springer*, 2009, pp. 680-691.

[117]. MohamedMAbd-Eldayem, EhabTAlnfrawy,andAlyAFahmya, Addition-Subtraction Chain for 160-bit Integers by using 2's Complex N Cruz-Cortés, F Rodríguez-Henríquez and C A Coello-Coello, "Addition Chain Length Minimization With Evolutionary Programming", *Proceedings of Genetic and Evolutionary Computation Conference (GECCO) ACM digital Library*, 2011.

[118]. S Dominguez-Isidro and E Mezura-Montes, "An Evolutionary Programming Algorithm to Find Minimal Addition Chains", *I Congreso Internacionalde Ingenieria Electronica, Instrumentacion y Computacion, de Juniodel, Minatitlan Veracruz, Mexico*, 2011.

[119]. Maurice Mignotte, "A Note on Addition Chains", International Journal of Algebra, 5(6), 2011.

[120]. Neil Michael Clift, "Calculating Optimal Addition Chains", *Journal of Computing, Springer*, 2011, pp. 265–284.

[121]. Arturo Rodriguez-Cristerna and Jose Torres-Jimenez, "A Genetic Algorithm for the Problem of Minimal Brauer Chains for Large Exponents", *Soft Computing Applications in Optimization, Control, and Recognition, Springer*, 2013.

[122]. K. Mani, "Generation of Addition Chain using Deterministic Division Based Method", *International Journal of Computer Science & Engineering Technology*, 4(05) (2013), pp. 553- 560.

[123]. P Anuradha Kameswari and B Ravitheja, Addition Chain For Lucas Sequences With Fast Computation Method, *International Journal of Applied Engineering Research*, 13(11) (2018), pp. 9413–9419.

[124]. Stjepanpicek, Carlos A Coello Coello, Domagojjakobovic and n elementens, Finding Short And Implementation - Friendly Addition Chains with Evolutionary Algorithms, *Journal of Heuristics*, 24, 2018, 457-481.

[125]. Aaron Hutchinson and koraykarabina, Constructing Multidimensional Differential Addition Chains and their applications, Springer, *Journal of Cryptographic Engineering*, 9, 2019, 1- 19.

[126]. Dustin Moody and Amadou Tall, On Addition-Subtraction Chains of Numbers With Low Hamming Weight", *Number Theory Mathematics*, 25, 2019, 155-168.

[127]. Hazem M. Bahig and Yasser Kotb, An Efficient Multicore Algorithm for Minimal Length Addition Chains, *Computers*, *MDBI*, 8, 2019.

[128]. Narendra Mohan, Lifetime Enhancement of Sensor Nodes Based on Optimized Sink Node Placement Approach, *International Journal of Engineering Trends and Technology*, 68.10, 2020, pp. 10-23.

APPENDIX - A.1

GENERATION OF E_{539038}(17,7) POINTS

Points from 1 to 10000 for A: 17, B: 7, P: 539039

(1,5) (1,539034), (2,7) (2,539032), (7,209384) (7,329655), (10,254036) (10,285003), (23,192123) (23,346916), (24,207140) (24,331899), (28,187853) (28,351186), (29,52806) (29,486233), (40,189675) (40,349364), (41,167198) (41,371841), (42,146230) (42,392809), (45,86800) (45,452239), (46,82678) (46,456361), (49,31538) (49,507501), (50,101736) (50,437303), (51,203954) (51,335085), (54,136270) (54,402769), (56,60393) (56,478646), (57,205207) (57,333832), (61,199701) (61,339338), (62,137992) (62,401047), (64,219222) (64,319817), (69,75066) (69,463973), (72,158175) (72,380864), (73,3422) (73,535617), (74,184001) (74,355038), (75,130341) (75,408698), (76,191752) (76,347287), (79,116748) (79,422291), (81,130515) (81,408524), (83,208693) (83,330346), (85,125993) (85,413046), (86,232781) (86,306258), (87,237349) (87,301690), (91,205623) (91,333416), (94,75784) (94,463255), (97,65630) (97,473409), (98,29893) (98,509146), (100,155858) (100,383181), (104,243678) (104,295361), (106,181812) (106,357227), (108,258954) (108,280085), (109,212549) (109,326490), (113,78317) (113,460722), (114,109103) (114,429936), (120,238388) (120,300651), (121,172234) (121,366805), (123,55452) (123,483587), (124,190117) (124,348922), (125,247077) (125,291962), (127,241155) (127,297884), (128,30395) (128,508644), (132,48252) (132,490787), (133,201326) (133,337713), (136,30090) (136,508949), (137,195904) (137,343135), (138,19228) (138,519811), (141,1675) (141,537364), (143,182822) (143,356217), (144,114930) (144,424109), (150,149132) (150,389907), (152,61587) (152,477452), (153,144363) (153,394676), (155,224959) (155,314080), (157,224524) (157,314515), (162,232469) (162,306570), (163,10691) (163,528348), (164,180150) (164,358889), (165,97469) (165,441570), (168,87115) (168,451924), (171,57970) (171,481069), (172,179701) (172,359338), (173,16639) (173,522400),

(174,202726) (174,336313), (175,51346) (175,487693), (176,154894) (176,384145), (180,32186) (180,506853), (182,223435) (182,315604), (184,134759) (184,404280), (189,233512) (189,305527), (190,149620) (190,389419), (191,192350) (191,346689), (193,172667) (193,366372), (197,65067) (197,473972), (201,62121) (201,476918), (202,198823) (202,340216), (203,100626) (203,438413), (204,106115) (204,432924), (205,221304) (205,317735), (206,169311) (206,369728), (207,67870) (207,471169), (211,102927) (211,436112), (212,75403) (212,463636), (218,193638) (218,345401), (219,68844) (219,470195), (220,194479) (220,344560), (221,59506) (221,479533), (222,66841) (222,472198), (223,22755) (223,516284), (224,260244) (224,278795), (228,5734) (228,533305), (229,107057) (229,431982), (232,145313) (232,393726), (233,246627) (233,292412), (235,118285) (235,420754), (236,169677) (236,369362), (239,6330) (239,532709), (240,102442) (240,436597), (241,72093) (241,466946), (242,2468) (242,536571), (243,225094) (243,313945), (247,254933) (247,284106), (251,64802) (251,474237), (252,221834) (252,317205), (254,120949) (254,418090), (255,87053) (255,451986), (260,27032) (260,512007), (261,42867) (261,496172), (262,146468) (262,392571), (263,184694) (263,354345), (264,237008) (264,302031), (270,91561) (270,447478), (272,142961) (272,396078), (276,3116) (276,535923), (277,106770) (277,432269), (279,61380) (279,477659), (280,36435) (280,502604), (284,212558) (284,326481), (286,171819) (286,367220), (295,230570) (295,308469), (298,267462) (298,271577), (302,184677) (302,354362), (304,123733) (304,415306), (305,115445) (305,423594), (306,179223) (306,359816), (310,237584) (310,301455), (311,56226) (311,482813), (312,180041) (312,358998), (313,92639) (313,446400), (314,14358) (314,524681), (317,138784) (317,400255), (318,35672) (318,503367), (319,12136) (319,526903), (323,75958) (323,463081), (324,27007) (324,512032), (326,69761) (326,469278), (327,255366) (327,283673), (328,101988) (328,437051), (330,266829) (330,272210), (332,237731) (332,301308), (334,184855) (334,354184), (335,118546) (335,420493), (336,129177) (336,409862), (339,242058) (339,296981),

(340,100611) (340,438428), (344,157661) (344,381378), (349,137117) (349,401922), (350,121292) (350,417747), (351,239257) (351,299782), (352,222740) (352,316299), (354,190475) (354,348564), (356,40896) (356,498143), (357,242472) (357,296567), (358,215919) (358,323120), (360,85588) (360,453451), (361,98347) (361,440692), (368,25300) (368,513739), (369,233031) (369,306008), (371,213319) (371,325720), (377,108669) (377,430370), (378,215453) (378,323586), (379,61576) (379,477463), (380,216107) (380,322932), (382,78488) (382,460551), (383,92068) (383,446971), (385,88600) (385,450439), (388,241302) (388,297737), (390,218017) (390,321022), (393,64411) (393,474628), (397,247680) (397,291359), (400,4911) (400,534128), (403,83202) (403,455837), (404,145098) (404,393941), (406,109020) (406,430019). (407,153750) (407,385289), (408,67017) (408,472022), (415,8063) (415,530976), (416,213816) (416,325223), (417,253992) (417,285047), (418,143225) (418,395814), (419,166201) (419,372838), (423,48169) (423,490870), (424,128734) (424,410305), (426,38322) (426,500717), (427,179656) (427,359383), (428,3239) (428,535800), (429,179231) (429,359808), (432,78296) (432,460743), (436,120266) (436,418773), (439,221120) (439,317919), (440,145207) (440,393832), (441,205971) (441,333068), (442,55761) (442,483278), (450,158669) (450,380370), (451,30005) (451,509034), (452,237940) (452,301099), (455,28875) (455,510164), (456,178525) (456,360514), (457,1810) (457,537229), (460,60648) (460,478391), (461,260512) (461,278527), (463,99537) (463,439502), (465,12341) (465,526698), (467,156012) (467,383027), (468,47355) (468,491684), (470,25227) (470,513812), (471,199881) (471,339158), (472,186527) (472,352512), (474,53180) (474,485859), (478,250839) (478,288200), (479,13476) (479,525563), (480,8409) (480,530630), (484,143655) (484,395384), (485,129775) (485,409264), (486,260299) (486,278740), (488,171598) (488,367441), (490,207312) (490,331727), (494,66369) (494,472670), (495,60885) (495,478154), (497,199804) (497,339235), (498,61309) (498,477730), (499,114427) (499,424612), (500,265589) (500,273450), (501,105829) (501,433210), (503,85369) (503,453670),

(506,137857) (506,401182), (509,33562) (509,505477), (510,268841) (510,270198), (513,37969) (513,501070), (517,169877) (517,369162), (518,72186) (518,466853), (522,239963) (522,299076), (525,160534) (525,378505), (528,99263) (528,439776), (529,264256) (529,274783), (530,175208) (530,363831), (531,126398) (531,412641), (532,174760) (532,364279), (534,222904) (534,316135), (535,26726) (535,512313), (536,15325) (536,523714), (537,213058) (537,325981), (539,31067) (539,507972), (540,190831) (540,348208), (543,36548) (543,502491), (544,202000) (544,337039), (545,103821) (545,435218), (546,92863) (546,446176), (547,160739) (547,378300), (555,102288) (555,436751), (561,33865) (561,505174**),** (562,47407) (562,491632), (566,143565) (566,395474), (567,45444) (567,493595), (568,139369) (568,399670), (572,112935) (572,426104), (574,124447) (574,414592), (577,119494) (577,419545), (578,118071) (578,420968), (581,104255) (581,434784), (582,46728) (582,492311), (585,252019) (585,287020), (586,85754) (586,453285), (587,190704) (587,348335), (588,111230) (588,427809), (589,47740) (589,491299), (591,100892) (591,438147), (592,216405) (592,322634), (595,77114) (595,461925), (596,211814) (596,327225), (597,68196) (597,470843), (603,90311) (603,448728), (604,87793) (604,451246), (607,101148) (607,437891), (608,17727) (608,521312), (610,24539) (610,514500), (611,244172) (611,294867), (612,100636) (612,438403), (613,198922) (613,340117), (620,158036) (620,381003), (621,137718) (621,401321), (626,73258) (626,465781), (629,162040) (629,376999), (630,65717) (630,473322), (633,69184) (633,469855), (638,171427) (638,367612), (640,19664) (640,519375), (641,110592) (641,428447), (642,143504) (642,395535), (643,103605) (643,435434), (645,36811) (645,502228), (646,257391) (646,281648), (647,235705) (647,303334), (648,205247) (648,333792), (653,86764) (653,452275), (654,32570) (654,506469), (656,186460) (656,352579), (660,51995) (660,487044), (661,264097) (661,274942), (663,87062) (663,451977), (666,30352) (666,508687), (668,114917) (668,424122), (670,8530) (670,530509), (672,48734) (672,490305), (674,114734) (674,424305), (677,131510) (677,407529),

(678,191429) (678,347610), (679,170500) (679,368539), (680,135868) (680,403171), (682,22432) (682,516607), (683,152938) (683,386101), (685,148031) (685,391008), (686,194138) (686,344901), (689,196374) (689,342665), (690,20785) (690,518254), (691,75992) (691,463047), (692,67325) (692,471714), (693,72091) (693,466948), (694,71202) (694,467837), (696,92215) (696,446824), (700,135407) (700,403632), (703,104143) (703,434896), (704,236083) (704,302956), (705,219536) (705,319503), (707,118065) (707,420974), (710,94836) (710,444203), (711,147919) (711,391120), (719,104716) (719,434323), (720,146799) (720,392240), (722,3761) (722,535278), (726,260689) (726,278350), (728,9482) (728,529557), (729,154353) (729,384686), (733,83180) (733,455859), (735,263674) (735,275365), (737,99327) (737,439712), (738,229775) (738,309264), (739,109140) (739,429899), (741,180432) (741,358607), (745,128463) (745,410576), (746,147341) (746,391698), (750,188301) (750,350738), (751,108836) (751,430203), (752,49941) (752,489098), (753,211612) (753,327427), (754,47327) (754,491712), (755,120978) (755,418061), (756,178124) (756,360915), (762,212822) (762,326217), (765,84603) (765,454436), (766,170508) (766,368531), (767,55460) (767,483579), (769,202260) (769,336779), (770,99761) (770,439278), (771,121378) (771,417661), (772,65535) (772,473504), (773,213165) (773,325874), (775,134699) (775,404340), (776,102369) (776,436670), (777,41521) (777,497518), (779,205963) (779,333076), (780,43673) (780,495366), (786,197383) (786,341656), (788,52054) (788,486985), (792,146691) (792,392348), (796,118218) (796,420821), (797,260061) (797,278978), (798,109261) (798,429778), (801,194696) (801,344343), (802,210401) (802,328638), (808,204249) (808,334790), (810,235582) (810,303457), (811,169485) (811,369554), (812,109203) (812,429836), (813,22521) (813,516518), (815,44522) (815,494517), (818,78940) (818,460099), (819,163460) (819,375579), (820,267239) (820,271800), (822,193329) (822,345710), (824,54612) (824,484427), (825,237614) (825,301425), (829,250463) (829,288576), (832,210367) (832,328672), (834,201978) (834,337061), (835,259647) (835,279392), (837,256918) (837,282121),

(838,89670) (838,449369), (839,58940) (839,480099), (840,192250) (840,346789), (841,76889) (841,462150), (842,92054) (842,446985), (843,61231) (843,477808), (844,265914) (844,273125), (845,262398) (845,276641), (847,160068) (847,378971), (848,148122) (848,390917), (849,155136) (849,383903), (850,103277) (850,435762), (852,36247) (852,502792), (853,169712) (853,369327), (855,97765) (855,441274), (857,75845) (857,463194), (860,117018) (860,422021), (866,234098) (866,304941), (868,105783) (868,433256), (869,142748) (869,396291), (871,164927) (871,374112), (872,3678) (872,535361), (874,214258) (874,324781), (880,219697) (880,319342), (882,76133) (882,462906), (883,260233) (883,278806), (885,103033) (885,436006), (887,34707) (887,504332), (888,266091) (888,272948), (889,269191) (889,269848), (890,254570) (890,284469), (891,81135) (891,457904), (892,75016) (892,464023), (894,178018) (894,361021), (895,171152) (895,367887), (897,226535) (897,312504), (898,51876) (898,487163), (900,238136) (900,300903), (903,174724) (903,364315), (904,121443) (904,417596), (907,174604) (907,364435), (908,165550) (908,373489), (911,142788) (911,396251), (912,142218) (912,396821), (913,71186) (913,467853), (914,35436) (914,503603), (915,46794) (915,492245), (923,161947) (923,377092), (925,6413) (925,532626), (927,57954) (927,481085), (928,92920) (928,446119), (929,207307) (929,331732), (930,92765) (930,446274), (932,63123) (932,475916), (933,267802) (933,271237), (937,193848) (937,345191), (941,199526) (941,339513), (944,102828) (944,436211), (947,133858) (947,405181), (948,176583) (948,362456), (949,187749) (949,351290), (950,36107) (950,502932), (955,120302) (955,418737), (960,22894) (960,516145), (962,54249) (962,484790), (966,30212) (966,508827), (967,73219) (967,465820), (968,263141) (968,275898), (969,147819) (969,391220), (970,229253) (970,309786), (972,125198) (972,413841), (974,51536) (974,487503), (977,18906) (977,520133), (983,76095) (983,462944), (984,231058) (984,307981), (985,156445) (985,382594), (987,237834) (987,301205), (988,18793) (988,520246), (993,74068) (993,464971), (997,230633) (997,308406), (998,205292) (998,333747),

(999,93901) (999,445138), (1000,160226)(1000,378813), (1001,222891) (1001,316148), (1003,197185)(1003,341854), (1004,142918)(1004,396121), (1005,41083)(1005,497956) (1006,141106)(1006,397933), (1011,62910)(1011,476129), (1012,24359) (1012,514680) (1014,52366)(1014,486673), (1015,41900)(1015,497139), (1017,260286)(1017,278753) (1018,97615)(1018,441424), (1020,132095)(1020,406944), (1021,28582)(1021,510457) (1023,251612)(1023,287427),(1024,215265)(1024,323774),(1025,210197)(1025,328842) (1026,31362)(1026,507677),(1027,266379)(1027,272660), (1028,237871)(1028,301168), (1030,22126)(1030,516913),(1033,127348)(1033,411691),(1035,137725)(1035,401314), (1039,225105)(1039,313934),(1042,90374)(1042,448665),(1043,148483)(1043,390556), (1044,265515)(1044,273524), (1045,204107)(1045,334932), (1048,5210)(1048,533829), (1050,78595)(1050,460444), (1051,38952)(1051,500087), (1055,223241) (1055,315798), (1056,219350)(1056,319689),(1059,242048)(1059,296991),(1060,121292)(1060,417747) (1065,166597)(1065,372442),(1067,116775)(1067,422264),(1068,26664)(1068,512375), (1069,211919)(1069,327120), (1071,6691)(1071,532348), (1073,187085)(1073,351954), (1074,41812)(1074,497227), (1076,29157) (1076,509882), (1077,43600)(1077,495439), (1078,2081) (1078,536958), (1079,74983) (1079,464056), (1080,83139) (1080,455900), (1082,172293)(1082,366746),(1083,45718)(1083,493321), (1086,159933)(1086,379106) (1088,154347)(1088,384692),(1089,123911)(1089,415128),(1090,114768)(1090,424271) (1091,85637)(1091,453402), (1092,193629)(1092,345410), (1093,223463)(1093,315576) (1094,175423)(1094,363616),(1097,120589)(1097,418450),(1098,116623)(1098,422416) (1099,26426)(1099,512613), (1101,134283)(1101,404756), (1103,214704)(1103,324335) (1105,240672)(1105,298367), (1107,36722)(1107,502317), (1108,64331)(1108,474708), (1109,16796)(1109,522243), (1111,78761)(1111,460278), (1112,127956)(1112,411083), (1113,230142)(1113,308897), (1117,211466)(1117,327573), (1118,82113)(1118,456926) (1119,236455)(1119,302584), (1120,25053)(1120,513986), (1121,153229)(1121,385810) (1127,47380) (1127,491659), (1129,40312) (1129,498727), (1130,2302) (1130,536737), (1131,78829)(1131,460210), (1132,141509)(1132,397530), (1135,118852)(1135,420187)

(1136,268371)(1136,270668), (1137,165498)(1137,373541), (1138,17360)(1138,521679)
(1139,36631)(1139,502408), (1140,128277)(1140,410762), (1141,183960)(1141,355079)
(1142,266094)(1142,272945), (1145,13475)(1145,525564), (1149,62221)(1149,476818),
(1152,28390)(1152,510649), (1153,43588)(1153,495451), (1154,252813)(1154,286226),
(1156,37558)(1156,501481), (1158,216711)(1158,322328), (1159,45480)(1159,493559),
(1161,42553)(1161,496486),(1164,153753)(1164,385286), (1165,102549)(1165,436490)
(1166,135189)(1166,403850),(1167,161149)(1167,377890),(1170,172734)(1170,366305)
(1178,146312)(1178,392727),(1181,34373)(1181,504666),(1183,210308)(1183,328731),
(1184,115570)(1184,423469),(1186,106232)(1186,432807),(1187,16346)(1187,522693),
(1188,123094)(1188,415945),(1189,104512)(1189,434527),(1190,149132)(1190,389907)
(1193,207374)(1193,331665),(1195,229155)(1195,309884),(1197,215615)(1197,323424)
(1199,88791)(1199,450248), (1200,11649)(1200,527390), (1205,101486)(1205,437553),
(1209,256362)(1209,282677), (1210,40264)(1210,498775), (1212,90639)(1212,448400),
(1214,154572)(1214,384467),(1217,181500)(1217,357539),(1218,202531)(1218,336508)
(1220,220877)(1220,318162),(1221,49284)(1221,489755), (1222,122091)(1222,416948),
(1224,47987)(1224,491052), (1225,224017)(1225,315022), (1227,245461)(1227,293578)
(1229,189319)(1229,349720), (1230,180749)(1230,358290), (1233,2703)(1233,536336),
(1235,97187)(1235,441852), (1236,163435)(1236,375604), (1241,123586)(1241,415453)
(1245,118649)(1245,420390), (1246,75052)(1246,463987), (1248,212136)(1248,326903)
(1250,230406)(1250,308633), (1251,22415)(1251,516624), (1252,142486)(1252,396553)
(1254,2466)(1254,536573), (1255,152952)(1255,386087), (1256,45507)(1256,493532),
(1257,130202)(1257,408837),(1259,238903)(1259,300136),(1263,134927)(1263,404112)
(1264,149127)(1264,389912),(1265,168142)(1265,370897),(1266,135126)(1266,403913)
(1269,77714)(1269,461325), (1270,145476)(1270,393563), (1274,51546)(1274,487493),
(1275,196535)(1275,342504),(1276,197933)(1276,341106),(1277,163353)(1277,375686)
(1280,153370)(1280,385669),(1282,168085)(1282,370954),(1283,166521)(1283,372518)
(1284,112447)(1284,426592),(1285,210724)(1285,328315),(1286,264976)(1286,274063)

(1292,54007)(1292,485032),(1298,148287)(1298,390752), (1302,244585)(1302,294454),
(1304,204237)(1304,334802), (1306,55277)(1306,483762), (1310,31137)(1310,507902),
(1311,121320)(1311,417719), (1313,107424)(1313,431615), (1314,45121)(1314,493918)
(1319,1820)(1319,537219), (1320,119077)(1320,419962), (1323,199749)(1323,339290)
(1324,39097)(1324,499942), (1325,5320)(1325,533719), (1326,269385)(1326,269654),
(1332,6327)(1332,532712), (1334,261059)(1334,277980), (1337,214614)(1337,324425)
(1339,162815)(1339,376224),(1340,186909)(1340,352130), (1342,40272)(1342,498767)
(1343,179724)(1343,359315),(1344,255159)(1344,283880), 1347,208441)(1347,330598)
(1349,108748)(1349,430291), (1351,62452)(1351,476587), (1353,27025)(1353,512014),
(1355,245102)(1355,293937),(1356,146074)(1356,392965),(1358,204103)(1358,334936)
(1361,45270)(1361,493769), (1362,254763)(1362,284276), (1363,147391)(1363,391648)
(1368,203311)(1368,335728), (1369,143092)(1369,395947), (1371,30597)(1371,508442)
(1372,96496)(1372,442543), (1374,73262)(1374,465777), (1376,54490)(1376,484549)
(1377,98384)(1377,440655), (1378,7464)(1378,531575), (1380,59163)(1380,479876)
(1383,209172)(1383,329867),(1386,195471)(1386,343568),(1387,193941)(1387,345098)
(1388,63208)(1388,475831), (1393,32534)(1393,506505), (1394,245569)(1394,293470)
(1395,214133)(1395,324906),(1398,218055)(1398,320984),(1399,266042)(1399,272997)
(1400,138045)(1400,400994), (1401,71660)(1401,467379), (1402,138839)(1402,400200)
(1403,38567)(1403,500472), (1404,259585)(1404,279454), (1407,76537)(1407,462502)
(1408,4240) (1408,534799), (1409,88193) (1409,450846), (1410,260373)(1410,278666)
(1413,268367)(1413,270672), (1414,208367)(1414,330672), (1417,98607)(1417,440432)
(1420,229128)(1420,309911), (1421,79524)(1421,459515), (1423,218821)(1423,320218)
(1427,182464)(1427,356575), (1428,124426)(1428,414613), (1429,86206)(1429,452833)
(1431,251520)(1431,287519),(1433,104266)(1433,434773),(1435,222298)(1435,316741)
(1437,36370)(1437,502669), (1441,104573)(1441,434466), (1442,226088)(1442,312951)
(1444,198244)(1444,340795), (1445,180400)(1445,358639), (1446,13338)(1446,525701)
(1447,185425)(1447,353614), (1450,50695)(1450,488344), (1451,260952)(1451,278087)

(1454,65843)(1454,473196), (1457,64917)(1457,474122), (1459,205850)(1459,333189)
(1462,224344)(1462,314695),(1469,173968)(1469,365071),(1473,214637)(1473,324402)
(1475,92550)(1475,446489), (1476,158149)(1476,380890), (1477,182165)(1477,356874)
(1478,196970)(1478,342069), (1479,181543)(1479,357496), (1480,72815)(1480,466224)
(1482,75830)(1482,463209), (1483,184583)(1483,354456), (1485,81188) (1485,457851)
(1487,95749)(1487,443290), (1489,170088)(1489,368951), (1491,126226)(1491,412813)
(1494,227773)(1494,311266),(1495,119443)(1495,419596),(1496,165259)(1496,373780)
(1497,224485)(1497,314554), (1498,52597)(1498,486442), (1501,234752)(1501,304287)
(1508,27821)(1508,511218), (1512,66304)(1512,472735), (1513,89395)(1513,449644)
(1516,155702)(1516,383337), (1518,45905)(1518,493134), (1520,104969)(1520,434070)
(1522,68105)(1522,470934), (1523,25829)(1523,513210), (1524,167778) (1524,371261)
(1526,42262)(1526,496777), (1527,180794)(1527,358245), (1529,55873) (1529,483166)
(1530,193506)(1530,345533),(1531,208197)(1531,330842),(1533,172722)(1533,366317)
(1535,101801)(1535,437238), (1538,20898)(1538,518141), (1543,94298) (1543,444741)
(1545,201360)(1545,337679),(1548,167294)(1548,371745),(1550,146142)(1550,392897)
(1551,221992)(1551,317047), (1553,257605)(1553,281434), (1555,98905)(1555,440134)
(1556,42986)(1556,496053), (1557,218330)(1557,320709), (1558,41304) (1558,497735)
(1560,157947)(1560,381092), (1561,203805)(1561,335234), (1563,43359)(1563,495680)
(1566,24056)(1566,514983), (1567,47304)(1567,491735), (1570,223411)(1570,315628)
(1573,41594)(1573,497445), (1574,26899)(1574,512140), (1575,245674)(1575,293365)
(1577,41976)(1577,497063), (1582,44955)(1582,494084), (1584,24218)(1584,514821)
(1585,165939)(1585,373100),(1586,143162)(1586,395877),(1588,119957)(1588,419082)
(1589,63702)(1589,475337),(1590,237149)(1590,301890), (1592,209882)(1592,329157)
(1594,149389)(1594,389650), (1596,190490)(1596,348549),1600,186023)(1600,353016)
(1604,171114)(1604,367925), (1605,95024)(1605,444015), (1613,14415)(1613,524624)
(1618,253588)(1618,285451),(1619,246778)(1619,292261),(1621,240478)(1621,298561)
(1623,70019)(1623,469020), (1624,99170)(1624,439869), (1625,83867)(1625,455172)

(1631,164415)(1631,374624),(1632,121528)(1632,417511),(1637,197266)(1637,341773)
(1639,57305)(1639,481734), (1641,77359)(1641,461680), (1643,66743)(1643,472296)
(1644,172279)(1644,366760), (1646,41286)(1646,497753), (1648,152942)(1648,386097)
(1649,267544)(1649,271495), (1652,56198)(1652,482841), (1653,96521)(1653,442518)
(1655,97969)(1655,441070), (1656,188833)(1656,350206), (1658,42964)(1658,496075)
(1659,4856)(1659,534183), (1660,249375)(1660,289664), (1661,1709)(1661,537330)
(1665,207552)(1665,331487), (1667,231863)(1667,307176), (1669,73382)(1669,465657)
(1674,236938)(1674,302101), (1677,19348)(1677,519691), (1678,54157)(1678,484882)
(1679,192176)(1679,346863),(1682,241594)(1682,297445),(1686,265124)(1686,273915)
(1687,103349)(1687,435690),(1688,125409)(1688,413630),(1690,232654)(1690,306385)
(1695,222767)(1695,316272), (1697,205199)(1697,333840), (1698,45752)(1698,493287)
(1699,98015)(1699,441024), (1700,262357)(1700,276682), (1702,93697)(1702,445342)
(1703,251011)(1703,288028), (1709,170263)(1709,368776), (1710,6239)(1710,532800)
(1713,171704)(1713,367335),(1714,163099)(1714,375940),(1715,173448)(1715,365591)
(1717,192519)(1717,346520), (1719,52757)(1719,486282), (1720,142437)(1720,396602)
(1721,127573)(1721,411466), (1723,12403)(1723,526636), (1725,36991)(1725,502048)
(1728,168358)(1728,370681), (1730,129729)(1730,409310), (1738,37461)(1738,501578)
(1739,28017)(1739,511022), (1740,27383)(1740,511656), (1743,107653)(1743,431386)
(1746,9809)(1746,529230), (1749,214020)(1749,325019), (1753,259877)(1753,279162)
(1759,17323)(1759,521716), (1764,209111)(1764,329928), (1765,19263)(1765,519776)
(1766,37247)(1766,501792), (1770,183165)(1770,355874), (1771,207440)(1771,331599)
(1773,28706)(1773,510333), (1774,146894)(1774,392145), (1775,138892)(1775,400147)
(1777,13110)(1777,525929), (1778,207385)(1778,331654), (1780,238549)(1780,300490)
(1781,177804)(1781,361235), (1782,212570)(1782,326469), (1784,35083)(1784,503956)
(1785,89677)(1785,449362), (1788,169316)(1788,369723), (1790,203138)(1790,335901)
(1792,141883)(1792,397156), (1796,14635)(1796,524404), (1798,16795)(1798,522244)
(1801,82535)(1801,456504), (1802,27855)(1802,511184), (1804,74807)(1804,464232)

(1805,97376)(1805,441663), (1806,105348)(1806,433691), (1807,149397)(1807,389642)
(1812,147936)(1812,391103), (1813,182221)(1813,356818), (1814,58725)(1814,480314)
(1817,246008)(1817,293031), (1818,62657)(1818,476382), (1822,85828)(1822,453211)
(1824,187748)(1824,351291), (1825,228489)(1825,310550), (1826,38787)(1826,500252)
(1835,249689)(1835,289350), (1836,211503)(1836,327536), (1837,45189)(1837,493850)
(1838,21532)(1838,517507), (1840,149593)(1840,389446), (1843,51782)(1843,487257)
(1844,204135)(1844,334904), (1845,20924)(1845,518115), (1850,225036)(1850,314003)
(1851,269489)(1851,269550), (1852,164084)(1852,374955), (1853,24783)(1853,514256)
(1855,73087)(1855,465952), (1857,16871)(1857,522168), (1858,115388)(1858,423651)
(1859,120735)(1859,418304),(1862,184312)(1862,354727),(1863,256628)(1863,282411)
(1869,202750)(1869,336289), (1871,15167)(1871,523872), (1873,159600)(1873,379439)
(1875,128326)(1875,410713), (1876,54248)(1876,484791), (1877,12220)(1877,526819)
(1878,170416)(1878,368623),(1879,126128)(1879,412911),(1882,248456)(1882,290583)
(1883,86582)(1883,452457), (1884,232339)(1884,306700), (1885,34892)(1885,504147)
(1886,173826)(1886,365213), (1888,171007)(1888,368032), (1892,19418)(1892,519621)
(1894,154036)(1894,385003),(1896,169850)(1896,369189),(1898,156747)(1898,382292)
(1900,98260)(1900,440779), (1901,266285)(1901,272754), (1902,117640)(1902,421399)
(1904,137606)(1904,401433), (1910,187194)(1910,351845), (1912,11888)(1912,527151)
(1913,57326)(1913,481713), (1914,263525)(1914,275514), (1915,42913)(1915,496126)
(1918,160146)(1918,378893),(1919,130668)(1919,408371),(1920,122451)(1920,416588)
(1921,56586)(1921,482453), (1922,253095)(1922,285944), (1924,228776)(1924,310263)
(1926,103023)(1926,436016), (1927,210220)(1927,328819), (1930,48617)(1930,490422)
(1931,9812)(1931,529227), (1933,241815)(1933,297224), (1934,150393)(1934,388646)
(1935,140386)(1935,398653),(1937,189055)(1937,349984),(1940,267198)(1940,271841)
(1941,164115)(1941,374924), (1944,130108)(1944,408931), (1945,36461)(1945,502578)
(1946,217885)(1946,321154),(1948,244771)(1948,294268),(1950,246296)(1950,292743)
(1951,74534)(1951,464505), (1954,10216)(1954,528823), (1959,207421)(1959,331618)

(1960,224301)(1960,314738), (1962,52454)(1962,486585), (1964,112099)(1964,426940)
(1965,188291)(1965,350748), (1966,58649)(1966,480390), (1967,9365)(1967,529674)
(1968,184331)(1968,354708),(1972,258546)(1972,280493),(1977,110117)(1977,428922)
(1978,205811)(1978,333228), (1979,49052)(1979,489987), (1981,74914)(1981,464125)
(1983,48955)(1983,490084), (1984,198539)(1984,340500), (1985,18146)(1985,520893)
(1986,115611)(1986,423428), (1987,74218)(1987,464821), (1989,54537)(1989,484502)
(1992,122588)(1992,416451),(1993,197319)(1993,341720),(1995,227044)(1995,311995)
(1996,102961)(1996,436078), (1998,661)(1998,538378), (1999,267753)(1999,271286)
(2002,1777)(2002,537262), (2003,233755)(2003,305284), (2006,116402)(2006,422637)
(2007,98791)(2007,440248), (2008,168465)(2008,370574), (2010,128888)(2010,410151)
(2015,190096)(2015,348943),(2016,263417)(2016,275622),(2017,105251)(2017,433788)
(2019,133793)(2019,405246),(2021,239084)(2021,299955),(2024,182417)(2024,356622)
(2027,254886)(2027,284153), (2030,50841)(2030,488198), (2033,220067)(2033,318972)
(2034,56852)(2034,482187), (2036,58419)(2036,480620), (2037,38293)(2037,500746)
(2038,38350)(2038,500689), (2043,249764)(2043,289275), (2045,208178)(2045,330861)
(2050,7135)(2050,531904), (2052,186468)(2052,352571), (2055,69344)(2055,469695)
(2056,125240)(2056,413799), (2057,170375)(2057,368664), (2059,10737)(2059,528302)
(2060,235538)(2060,303501), (2062,20931)(2062,518108), (2063,119573)(2063,419466)
(2064,28531)(2064,510508), (2065,147258)(2065,391781), (2067,31385)(2067,507654)
(2070,243817)(2070,295222), (2071,214659)(2071,324380), (2075,25961)(2075,513078)
(2077,144437)(2077,394602),(2079,169850)(2079,369189),(2081,268693)(2081,270346)
(2085,7739)(2085,531300), (2086,38152)(2086,500887), (2087,117628)(2087,421411)
(2088,71420)(2088,467619), (2090,147323)(2090,391716), (2091,150215)(2091,388824)
(2093,84459)(2093,454580), (2096,44639)(2096,494400), (2097,102459)(2097,436580)
(2098,167929)(2098,371110), (2099,20884),(2099,518155),(2100,245443)(2100,293596)
(2101,61926)(2101,477113), (2103,266481)(2103,272558), (2104,233794)(2104,305245)
(2105,49533)(2105,489506), (2111,142064)(2111,396975), (2113,221631)(2113,317408)

(2115,185429)(2115,353610), (2117,252145)(2117,286894), (2118,22053)(2118,516986)
(2121,107155)(2121,431884), (2122,37251)(2122,501788), (2124,188083)(2124,350956)
(2127,241460)(2127,297579), (2130,67324)(2130,471715), (2131,238219)(2131,300820)
(2133,266571)(2133,272468), (2141,251664)(2141,287375), (2142,45379)(2142,493660)
(2143,6475)(2143,532564), (2144,252905)(2144,286134), (2152,17479)(2152,521560)
(2154,223393)(2154,315646),(2156,104209)(2156,434830),(2159,222574)(2159,316465)
(2160,48821)(2160,490218), (2165,232064)(2165,306975), (2166,134291)(2166,404748)
(2167,115628)(2167,423411), (2171,228081)(2171,310958), (2172,16614)(2172,522425)
(2174,92446)(2174,446593), (2175,132765)(2175,406274), (2176,251550)(2176,287489)
(2177,164375)(2177,374664),(2178,149975)(2178,389064), 2184,231592)(2184,307447)
(2185,213043)(2185,325996), (2188,25510)(2188,513529), (2190,199882)(2190,339157)
(2191,2575)(2191,536464), (2194,149100)(2194,389939), (2195,127959)(2195,411080)
(2197,34888), (2197,504151), (2199,98247)(2199,440792), (2202,107141)(2202,431898)
(2203,96634)(2203,442405), (2208,183695)(2208,355344), (2212,257422)(2212,281617)
(2213,95250)(2213,443789), (2214,122736)(2214,416303), (2216,71341)(2216,467698)
(2218,41393)(2218,497646), (2219,93790)(2219,445249), (2220,259081)(2220,279958)
(2225,20803)(2225,518236), (2227,10331)(2227,528708), (2232,213764)(2232,325275)
(2234,102545)(2234,436494), (2236,100984)(2236,438055), (2239,84122)(2239,454917)
(2242,223262)(2242,315777), (2247,85731)(2247,453308), (2248,108703)(2248,430336)
(2252,226996)(2252,312043),(2253,240071)(2253,298968),(2254,156080)(2254,382959)
(2256,149723)(2256,389316), (2260,138839)(2260,400200), (2262,34408)(2262,504631)
(2264,34650)(2264,504389), (2266,234384)(2266,304655), (2267,229127)(2267,309912)
(2268,78763)(2268,460276), (2269,204827)(2269,334212), (2270,222596)(2270,316443)
(2271,199492)(2271,339547), (2273,90309)(2273,448730), (2274,159235)(2274,379804)
(2275,137417)(2275,401622), (2277,90238)(2277,448801), (2278,222457)(2278,316582)
(2280,119369)(2280,419670), (2282,5391)(2282,533648), (2283,204521)(2283,334518)
(2285,63621)(2285,475418), (2286,59311)(2286,479728), (2287,202190)(2287,336849)

(2289,234646)(2289,304393), (2291,22852)(2291,516187), (2294,67484)(2294,471555)
(2298,217492)(2298,321547), (2306,9680)(2306,529359), (2311,168740)(2311,370299)
(2312,115158)(2312,423881), (2314,182391)(2314,356648), (2316,3370)(2316,535669)
(2317,253223)(2317,285816), (2318,45584)(2318,493455), (2321,82598)(2321,456441)
(2328,87542)(2328,451497), (2329,90426)(2329,448613), (2331,55783)(2331,483256)
(2332,135253)(2332,403786), (2336,9496)(2336,529543), (2337,197145)(2337,341894)
(2338,151994)(2338,387045),(2340,153325)(2340,385714),(2347,200090)(2347,338949)
(2349,123321)(2349,415718),(2350,113475)(2350,425564),(2351,128674)(2351,410365)
(2352,195587)(2352,343452), (2353,85395)(2353,453644), (2355,1045)(2355,537994)
(2356,53505)(2356,485534), (2359,206377)(2359,332662), (2360,63865)(2360,475174)
(2365,77216)(2365,461823), (2368,158362)(2368,380677), (2369,261101)(2369,277938)
(2372,81335)(2372,457704), (2373,237060)(2373,301979), (2374,132727)(2374,406312)
(2376,124802)(2376,414237), (2377,131788)(2377,407251), (2378,80346)(2378,458693)
(2384,48067)(2384,490972), (2385,158531)(2385,380508), (2386,199890)(2386,339149)
(2387,54997)(2387,484042), (2388,91294)(2388,447745), (2391,258121)(2391,280918)
(2392,110929)(2392,428110),(2393,101910)(2393,437129),(2394,137115)(2394,401924)
(2397,251710)(2397,287329), (2399,97398)(2399,441641), (2403,44918)(2403,494121)
(2406,94829)(2406,444210), (2413,267513)(2413,271526), (2415,265094)(2415,273945)
(2416,102264)(2416,436775),(2417,153268)(2417,385771),(2418,214992)(2418,324047)
(2422,43624)(2422,495415), (2423,179735)(2423,359304), (2424,58172)(2424,480867)
(2427,89878)(2427,449161), (2429,141552)(2429,397487), (2431,160554)(2431,378485)
(2434,181331)(2434,357708), (2437,194614)(2437,344425), (2438,67447)(2438,471592)
(2439,43220)(2439,495819), (2441,91358)(2441,447681), (2442,124821)(2442,414218)
(2444,188555)(2444,350484), (2445,23130)(2445,515909), (2450,84554)(2450,454485)
(2451,25860)(2451,513179), (2452,62225)(2452,476814), (2454,36237)(2454,502802)
(2456,61467)(2456,477572), (2458,158297)(2458,380742), (2459,204966)(2459,334073)
(2462,163562)(2462,375477),(2464,134801)(2464,404238),(2466,152474)(2466,386565)

(2467,21012)(2467,518027), (2468,21505)(2468,517534), (2469,119477)(2469,419562)
(2471,155522)(2471,383517),(2472,236632)(2472,302407),(2475,148094)(2475,390945)
(2476,207809)(2476,331230),(2480,158923)(2480,380116),(2482,250459)(2482,288580)
(2483,203678)(2483,335361),(2484,186544)(2484,352495),(2486,194663)(2486,344376)
(2487,95267)(2487,443772), (2489,103046)(2489,435993), (2490,122925)(2490,416114)
(2492,238734)(2492,300305),(2497,172864)(2497,366175),(2499,202749)(2499,336290)
(2500,84623)(2500,454416), (2501,231965)(2501,307074), (2502,179554)(2502,359485)
(2507,197360)(2507,341679), (2509,41461)(2509,497578), (2511,255022)(2511,284017)
(2512,173189)(2512,365850), (2515,266037)(2515,273002), (2518,4495)(2518,534544)
(2523,130792)(2523,408247), (2524,48165)(2524,490874), (2527,154781)(2527,384258)
(2528,95777)(2528,443262), (2531,161313)(2531,377726), (2532,71229)(2532,467810)
(2533,77696)(2533,461343), (2535,232838)(2535,306201), (2537,63655)(2537,475384)
(2539,264253)(2539,274786), (2542,90077)(2542,448962), (2544,254098)(2544,284941)
(2546,32384)(2546,506655), (2547,213898)(2547,325141), (2550,141935)(2550,397104)
(2552,36684)(2552,502355), (2553,210043)(2553,328996), (2555,35632)(2555,503407)
(2559,21240)(2559,517799), (2560,227400)(2560,311639), (2562,224944)(2562,314095)
(2564,55496)(2564,483543), (2565,73107) (2565,465932), (2567,467) (2567,538572)
(2569,193942)(2569,345097), (2571,31048)(2571,507991), (2575,23571)(2575,515468)
(2578,124045)(2578,414994), (2581,11318)(2581,527721), (2591,235306)(2591,303733)
(2596,64538)(2596,474501), (2597,112649)(2597,426390), (2598,40614)(2598,498425)
(2600,136072)(2600,402967), (2601,53691)(2601,485348), (2602,11714)(2602,527325)
(2606,122708)(2606,416331), (2607,80889)(2607,458150), (2608,126450)(2608,412589)
(2609,110058)(2609,428981), (2610,13929)(2610,525110), (2620,228964)(2620,310075)
(2621,171661)(2621,367378),(2622,222224)(2622,316815),(2624,122387)(2624,416652)
(2626,147969)(2626,391070), (2627,7067)(2627,531972), (2629,268412)(2629,270627)
(2630,207984)(2630,331055),(2633,123660)(2633,415379),(2634,264936)(2634,274103)
(2637,235787)(2637,303252), (2639,3255)(2639,535784), (2640,12662)(2640,526377)

(2643,46521)(2643,492518), (2644,82324)(2644,456715), (2648,232224)(2648,306815)
(2652,13950)(2652,525089), (2653,4768)(2653,534271), (2654,21131) (2654,517908)
(2655,62370)(2655,476669), (2656,131496)(2656,407543), (2657,158283)(2657,380756)
(2659,181369)(2659,357670),(2660,107210)(2660,431829),(2663,138868)(2663,400171)
(2667,46321)(2667,492718), (2670,88309)(2670,450730), (2671,244885)(2671,294154)
(2673,1877)(2673,537162), (2675,211176)(2675,327863), (2677,263900)(2677,275139)
(2682,79424)(2682,459615), (2684,7796)(2684,531243), (2685,218841)(2685,320198)
(2686,105986)(2686,433053), (2693,656)(2693,538383), (2696,217058)(2696,321981)
(2699,26710)(2699,512329), (2701,63717)(2701,475322), (2702,250729)(2702,288310)
(2703,262490)(2703,276549), (2705,249750)(2705,289289), (2706,89749)(2706,449290)
(2707,240165)(2707,298874), (2712,66090)(2712,472949), (2713,266792)(2713,272247)
(2715,182730)(2715,356309),(2717,110320)(2717,428719),(2722,170168)(2722,368871)
(2725,247696)(2725,291343), (2726,31835)(2726,507204), (2727,250971)(2727,288068)
(2728,252815)(2728,286224), (2729,85395)(2729,453644), (2736,46238)(2736,492801)
(2737,127762)(2737,411277),(2738,266126)(2738,272913),(2739,129227)(2739,409812)
(2740,37351)(2740,501688), (2742,81999)(2742,457040), (2743,124881)(2743,414158)
(2745,236455)(2745,302584),(2746,108773)(2746,430266),(2750,208922)(2750,330117)
(2751,137409)(2751,401630),(2753,175392)(2753,363647),(2755,187162)(2755,351877)
(2757,3432)(2757,535607), (2762,248171)(2762,290868), (2763,195917)(2763,343122)
(2767,150725)(2767,388314), (2768,149671)(2768,389368), (2769,40711)(2769,498328)
(2770,212346)(2770,326693), (2774,45694)(2774,493345), (2775,182098)(2775,356941)
(2778,144373)(2778,394666), (2779,138673)(2779,400366), (2781,1641)(2781,537398)
(2782,79242)(2782,459797), (2784,125014)(2784,414025), (2787,177354)(2787,361685)
(2788,114301)(2788,424738),(2791,123591)(2791,415448),(2793,235999)(2793,303040)
(2794,6855)(2794,532184), (2797,1894)(2797,537145), (2798,200054)(2798,338985)
(2800,39588)(2800,499451), (2802,11946)(2802,527093), (2803,152140)(2803,386899)
(2807,250955)(2807,288084), (2811,82004)(2811,457035), (2812,121808)(2812,417231)

(2814,57165)(2814,481874), (2816,132166)(2816,406873), (2818,105097)(2818,433942)
(2823,58262)(2823,480777), (2824,247526)(2824,291513), (2825,209550)(2825,329489)
(2828,147849)(2828,391190), (2829,52999)(2829,486040), (2830,189421)(2830,349618)
(2831,254566)(2831,284473), (2832,20645)(2832,518394), (2833,134109)(2833,404930)
(2834,243533)(2834,295506),(2835,164175)(2835,374864),(2837,154579)(2837,384460)
(2838,90010)(2838,449029), (2840,157374)(2840,381665), (2842,207990)(2842,331049)
(2844,236740)(2844,302299), (2847,96016)(2847,443023), (2849,45550)(2849,493489)
(2850,126727)(2850,412312), (2851,96161)(2851,442878), (2852,17463)(2852,521576)
(2853,188048)(2853,350991), (2854,92713)(2854,446326), (2855,101859)(2855,437180)
(2856,236802)(2856,302237),(2859,148033)(2859,391006),(2861,231164)(2861,307875)
(2862,113717)(2862,425322),(2868,213370)(2868,325669),(2869,146388)(2869,392651)
(2870,18159)(2870,520880), (2873,234677)(2873,304362), (2875,120386)(2875,418653)
(2877,79586)(2877,459453), (2879,3081)(2879,535958), (2880,260755)(2880,278284)
(2881,197641)(2881,341398), (2883,24826)(2883,514213), (2884,70420)(2884,468619)
(2885,133621)(2885,405418), (2887,127377)(2887,411662), (2889,10703)(2889,528336)
(2890,61742)(2890,477297), (2891,168905)(2891,370134), (2892,98135)(2892,440904)
(2894,172024)(2894,367015), (2895,91444)(2895,447595), (2896,53311)(2896,485728)
(2898,200628)(2898,338411), (2900,32895)(2900,506144), (2901,205873)(2901,333166)
(2903,15505)(2903,523534), (2906,101382)(2906,437657), (2907,106998)(2907,432041)
(2908,26985)(2908,512054), (2909,228413)(2909,310626), (2913,13787)(2913,525252)
(2919,147252)(2919,391787),(2921,221430)(2921,317609),(2922,203321)(2922,335718)
(2923,171579)(2923,367460),(2924,239010)(2924,300029),(2926,155103)(2926,383936)
(2927,185200)(2927,353839), (2928,32168)(2928,506871), (2930,152382)(2930,386657)
(2933,29291)(2933,509748), (2934,239406)(2934,299633), (2935,77618)(2935,461421)
(2936,198853)(2936,340186), (2940,13329)(2940,525710), (2944,36785)(2944,502254)
(2945,136553)(2945,402486), (2946,119673)(2946,419366), (2949,54081)(2949,484958)
(2950,55829)(2950,483210), (2952,106499)(2952,432540), (2953,226480)(2953,312559)

(2954,150094)(2954,388945), (2955,30642)(2955,508397), (2962,138026)(2962,401013)
(2963,233993)(2963,305046), (2964,43822)(2964,495217), (2965,175723)(2965,363316)
(2966,231510)(2966,307529), (2967,22262)(2967,516777), (2971,124492)(2971,414547)
(2972,46607)(2972,492432), (2973,228961)(2973,310078), (2974,79273)(2974,459766)
(2977,131482)(2977,407557), (2978,189536)(2978,349503), (2980,249)(2980,538790)
(2981,64394)(2981,474645), (2983,229946)(2983,309093), (2986,120567)(2986,418472)
(2987,231528)(2987,307511), (2992,69681)(2992,469358), (2993,103465)(2993,435574)
(2994,213470)(2994,325569), (2995,33486)(2995,505553), (2996,20703)(2996,518336)
(2997,256266)(2997,282773), (3000,28814)(3000,510225), (3001,185250)(3001,353789)
(3003,71638)(3003,467401), (3005,81717)(3005,457322), (3007,60032)(3007,479007)
(3009,135606)(3009,403433), (3010,51119)(3010,487920), (3011,11912)(3011,527127)
(3012,51617)(3012,487422), (3015,17989)(3015,521050), (3017,245077)(3017,293962)
(3020,130762)(3020,408277), (3021,52316)(3021,486723), (3023,171201)(3023,367838)
(3025,210974)(3025,328065),(3026,179039)(3026,360000),(3028,161407)(3028,377632)
(3029,267935)(3029,271104), (3031,33917)(3031,505122), (3034,210307)(3034,328732)
(3036,265307)(3036,273732),(3044,261736)(3044,277303),(3045,170815)(3045,368224)
(3046,236297)(3046,302742), (3047,229919)(3047,309120), (3049,21138)(3049,517901)
(3050,167015)(3050,372024),(3057,102005)(3057,437034),(3059,228564)(3059,310475)
(3060,79612)(3060,459427), (3063,101946)(3063,437093), (3066,118085)(3066,420954)
(3067,139235)(3067,399804),(3069,194595)(3069,344444),(3071,223183)(3071,315856)
(3072,225226)(3072,313813),(3079,120498)(3079,418541),(3081,119378)(3081,419661)
(3083,133356)(3083,405683), (3084,233895)(3084,305144), (3086,39439)(3086,499600)
(3087,156484)(3087,382555),(3088,100342)(3088,438697),(3089,111115)(3089,427924)
(3091,239971)(3091,299068), (3092,85527)(3092,453512), (3093,39873)(3093,499166)
(3094,101190)(3094,437849), (3096,2576)(3096,536463), (3098,134830)(3098,404209)
(3099,147422)(3099,391617),(3100,156674)(3100,382365),(3102,106501)(3102,432538)
(3104,215728)(3104,323311),(3109,143327)(3109,395712),(3112,205287)(3112,333752)

(3113,73497)(3113,465542), (3115,238545)(3115,300494), (3116,56098)(3116,482941)
(3121,64733)(3121,474306), (3122,206427)(3122,332612), (3123,152299)(3123,386740)
(3124,13509)(3124,525530), (3125,192553)(3125,346486), (3126,227574)(3126,311465)
(3129,233737)(3129,305302),(3131,215446)(3131,323593),(3132,259009)(3132,280030)
(3135,188722)(3135,350317), (3137,89024)(3137,450015), (3139,32231)(3139,506808)
(3140,84263)(3140,454776), (3142,35087)(3142,503952), (3144,256539)(3144,282500)
(3148,30273)(3148,508766), (3149,169179)(3149,369860), (3151,60167)(3151,478872)
(3153,4112)(3153,534927), (3156,155828)(3156,383211), (3159,204795)(3159,334244)
(3163,53899)(3163,485140), (3164,104191)(3164,434848), (3165,210887)(3165,328152)
(3166,209462)(3166,329577),(3168,212939)(3168,326100),(3171,103250)(3171,435789)
(3172,262527)(3172,276512), (3173,26267)(3173,512772), (3184,128342)(3184,410697)
(3185,103755)(3185,435284), (3190,237463)(3190,301576), (3191,31169)(3191,507870)
(3192,60743)(3192,478296), (3193,18584)(3193,520455), (3194,267843)(3194,271196)
(3195,220039)(3195,319000), (3199,6640)(3199,532399), (3201,86469)(3201,452570)
(3202,171978)(3202,367061), (3203,238897)(3203,300142), (3207,23014)(3207,516025)
(3209,80855)(3209,458184), (3210,132337)(3210,406702), (3211,231274)(3211,307765)
(3212,56448)(3212,482591), (3213,203940)(3213,335099), (3215,221133)(3215,317906)
(3216,16038)(3216,523001), (3217,64891)(3217,474148), (3218,46332)(3218,492707)
(3220,44778)(3220,494261), (3221,193719)(3221,345320), (3222,45021)(3222,494018)
(3224,223105)(3224,315934),(3228,115552)(3228,423487),(3229,124911)(3229,414128)
(3230,31421)(3230,507618), (3232,64118)(3232,474921), (3233,8006)(3233,531033)
(3236,211899)(3236,327140), (3237,55191)(3237,483848), (3240,137825)(3240,401214)
(3241,105265)(3241,433774), (3242,262414)(3242,276625), (3251,85831)(3251,453208)
(3253,82689)(3253,456350), (3254,1552)(3254,537487), (3255,100698)(3255,438341)
(3256,188458)(3256,350581),(3257,102009)(3257,437030),(3258,120487)(3258,418552)
(3260,127867)(3260,411172), (3263,25138)(3263,513901), (3264,205559)(3264,333480)
(3266,135961)(3266,403078),(3270,156402)(3270,382637),(3277,258799)(3277,280240)

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

A Survey on the Security Features of Cryptographic Techniques in Mobile Devices

Dr. Mani[1], A. Mullai [2]

Associate Professor, Department of Computer Science, Puthanampatti, Affiliated to Bharathidasan University, Trichy, Tamil Nadu, India[1]

PhD Research Scholar, Department of Computer Science, Puthanampatti, Affiliated to Bharathidasan University, Trichy, Tamil Nadu, India [2]

ABSTRACT: A revolution of technological advancement has taken place in the field of Communication Technology with the introduction of Wireless Mobile Devices by introducing multifunctional applications embedded in a small device by replacing the traditional and fixed wired technology. Mobile devices play a vital role in everyday life since they provide variety of ubiquitous services. In recent years, the availability of these devices and their ubiquitous services has increased significantly. This is because various forms of connectivity such as Global System for Mobile Communications (GSM), General Packet Radio Service (GPRS), Bluetooth and Wi-Fi (Wireless Fidelity) etc. are provided to them. Even though some challenges in mobile computing devices cater the needs of the users, the transmission of information can be done very easily, quickly without having any previous knowledge because they are user friendly. The information transmitted through the air by the mobile devices may sometimes being hacked by the hackers. To avoid hacking Security plays a vital role in transmitting the information and it can be achieved by using various cryptographic algorithms to prevent from such attacks. This paper gives a comprehensive survey of cryptographic algorithms and techniques which are being used in mobile devices.

KEYWORDS: Mobile Devices, challenges, hackers, Cryptographic algorithms, techniques.

I. INTRODUCTION

Mobile Computing Portable devices like laptop, palmtop etc. gives an easy access to the people with diverse sources of global information instantaneously anywhere at any place and at any time. It is a technology constantly developing towards the needs of human expectations by using the concept of Bring Your Own Device – Bring Your Own Technology (BYOD – BYOT). A mobile device may be a Personal Digital Assistant (PDA), a handy Cell phone or Web phone, a laptop, or any one of the above numerous devices that allow the user to complete the tasks without being tethered, or connected, to a network. The environment of wireless and mobile bring about different challenges to the users and service providers. The physical constraints like the weight of the device, the battery, the size of the screen, portability, quality of radio transmission, and error rates become more important. Even though the facility of the devices include the mobility of the user, the device, the network, the service provider and also some additional uncertainties, they give opportunities to the users the provision of new services and supplementary information. The major challenges in mobile computing are low bandwidth, high error rate, power restrictions, security, limited capabilities, disconnection and the problems created due to the mobility of the client. Inspite of these challenges security becomes a major concern, because they are connected anonymously. By the application of cryptographic algorithms in mobile computing, the hackers don't get the chance to access the mobile units. Various cryptographic algorithms have been used to maintain security in mobile devices and they provide confidentiality, integrity, availability, non-repudiation, authorization and trust and accounting (CIANATA). This paper gives an overview of various cryptographic techniques which are used to provide such security services in mobile devices.

II. RELATED WORKS

Mavridis I., Pangalos G. [1], in their paper, have discussed the operational and security issues of mobile components in distributed environments. Further they illustrated to eliminate the intrinsic problem of wireless

 DOI: 10.15680/IJIRCCE.2016. 0402186

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

networking using the mobile agents. They applied some security mechanism in their model which is to be implemented in a healthcare paradigm, with some special conditions.

In 2000, Erik Olson and Woojin Yu, [2] surveyed various symmetric key algorithms viz., RC5, RC6, Twofish, and Triple-DES and their usage in mobile computing, specifically in the Palm Pilot, which uses Motorola's Dragon Ball-EZ processor. They illustrated that the architecture used in the processor is similar to the 68K processor and it does not provide the power and versatility of current processors.

In 2000, Wendy Chou [3], surveyed the explosive growth in the usage of mobile and wireless devices demands a new generation of Public Key Cryptography (PKC) schemes, and the limitations on power, bandwidth to provide security in mobile devices, use of Elliptic Curve Cryptography (ECC), its security, performance and also its applications.

In 2002, Limor Elbaz [4], implemented PKC in security of wireless devices and the use of Public Key Infrastructure (PKI) in current as well as in the future applications of mobile phones. Further he showed that the Discretix Crypto Cell implementation of cryptographic algorithms which enable wireless devices to become PKI-enabled cum efficient, lightweight and standard-compliant.

In 2003, Dharma P. Agrawal et al.[5], discussed the technology in mobile computing users by combining wireless networking and mobility which serves anytime and anywhere with of various new applications and also services. They had also analyzed some security issues and various threats in the existing countermeasures. They concluded that encryption plays an important role for secured communication in mobile computing environments.

In 2006, Hanping Lufei and Weisong Shi [6], discussed the emergence of heterogeneous devices and diverse networks, and the difficulty in using a one-size-fits-all encryption algorithm. They also explained the deployment of encryption algorithms to choose an appropriate encryption algorithm from multiple algorithms based on the characteristics of heterogeneous mobile computing environments. They proposed an adaptive encryption protocol, to choose a proper encryption algorithm dynamically which enhances security from the candidate algorithms, and minimizes the time overhead.

In 2008, Abhishek Kumar Gupta [7], discussed the need for information as a driving force for the incoming growth in Web technology, wireless communication, and portable computing devices and also explained the field of mobile computing (computing and communication) with the aim of providing seamless computing environment for mobile users, which are all dependent on information and it is available only by accessing a network. Further they discussed that the mobility can also cause wireless connections to be lost or degraded when the users travel beyond the limitations of network transceivers or enter areas of high interference.

In [2009], S. Krishna Mohan Rao and Dr. A Venugopal Reddy [8], discussed Data dissemination in asymmetrical communication environment, where the capacity of the downlink communication is much greater than the uplink communication capacity and it is best suited for mobile environment. The important issue discussed in this paper is that the data dissemination which illustrates quickly access of the data item in mobile devices with minimum access time so that the mobile clients save the precious battery power while they are moving from one place to another.

In [2009], widespread growth in applications for resource-limited Wireless Sensor Networks (WSN), and also the need for reliable and efficient security mechanisms using two potential block ciphers, namely the RC5 and AES-Rijindael discussed and analyzed the suitability of the algorithm for resource-limited wireless network security by M. Razvi Doomun, and KMS Soyjaudah [9].

In [2009], Kar and Banshidhar Majhi[10], proposed an efficient password security of Multi-Party key exchange protocol based on elliptic curve discrete logarithm problem (ECDLP), and these protocols allow a group of parties communicating over a public network to establish a common secret key called Session Key and also build protocol for password authentication model, where group members were assumed to hold an individual password rather than a common password and two one-way hash functions to build the security level high.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

In [2009], Mooseeop Kim et.al. [11], proposed a compact architecture for a cryptographic engine on a mobile platform, which has very stringent limitations with respect to the circuit area and the consuming power .It is highly effective to implement the scalable RSA and unified SHA algorithms with a minimum resource usage. The combined performance results of circuit area, power efficiency, throughput, and functionality strongly indicate that the proposed architecture for cryptographic hardware is suitable for mobile computing systems.

In [2010], Bruno P.S. Rocha et. al [12], demonstrated a security service, which works as a middleware, to dynamically change the security protocols used between two peers and these changes can occur based on variations on wireless medium parameters, system resource usage, available hardware resources, application-defined Quality of Service (QoS) metrics, and desired data security levels. They provide the solution to some static security protocols and adaptability of middleware in different conditions of medium and system, and shows performance gain in the execution of cryptographic primitives, through the use of data semantics.

In [2010], Sathish Alampalayam Kumar [13], suggested a mobile agent based mobile computing system, the classification of various types of security attacks, the security solutions for those types of attacks proposed by various schemes and the open research issues in providing security for mobile agent based computing systems.

In [2011], Sameer Hasan et. al. [14], proposed a non-server (that is P2P) architecture PKC to secure the mobile communications. They have discussed and implemented various security services needed for mobile communication. Compared with server based architecture, this architecture has low risk and the security has been improved to avoid many attacks. They used NTRU algorithm for public key cryptography in non-server architecture and tested on real equipment, the solution security and potential risks.

In [2011], Rahat Afreen and S.C. Mehrotra [15], discussed the ECC emerged in its proper implementation in various directions to analyze in hardware as well as software platforms. Helena Rifa-Pous and Jordi Herrera-Joancomarti [16], discussed the performances of different cryptographic algorithms in PDAs and compared it with device's basic costs in terms of operating system, screen, and network interfaces to determine the overhead and the results were used to estimate the costs of network security protocols design.

In [2011], Jagdish Bhatta and Lok Prakash Pandey [17], proposed a software level cryptographic protocol implementations to measure the energy level through the device's serial port, running them and measuring their power consumption. The results show that the proposed cryptographic protocol provides a guaranteed better security and acquires very less consumption of energy than the existing cryptographic protocols. The performance analysis are compared and proved that the proposed scheme is to be more simple, secure and efficient.

In [2012], K. Sathish Kumar et. al. [18], explained the mobile hand-held device in an efficient way to deliver real time data to users. They designed and implemented an energy efficient authentication protocol that accomplishes a high level security with minimum energy consumption for mobile devices.

In [2012], Masoud Nosrati et. al. [19], proposed an algorithm for security mechanism in different types of mobile devices and the operation systems. This security mechanism uses some algorithms to scramble data into unreadable text which can be only decoded or decrypted by those who possess the associated key and these algorithms consume a significant amount of computing resources such as CPU time, memory, battery power and computation time.

In [2012], Ravinder Singh Mann et. al. [20], presented the comparative analysis of ECC, AES and RSA algorithms experimentally with parameters such as computation time and complexity of the algorithms. Based on the result it was concluded that ECC has more complexity when compared to AES and RSA in mobile devices.

In [2013], Giripunje et al. [21], discussed many differences in mobile devices, their capabilities, computational powers and security requirements in networking environments. The security of mobile communication is concerned with mobile confidentiality, authentication, integrity and non-repudiation. They have mentioned that the currently available network security mechanisms are inadequate. They provided effective security solution using PKC and its

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

implementation in two parts: first, design for API for ECC which generates shared key for secure communication and secondly, a web service is created which distributes this key to validate the mobile user.

In [2013], Ameya Nayak [22], discussed the growing android community, its malware attacks, security concerns, aid in serving as the continuous challenges of identifying current, future vulnerabilities as well as incorporating security strategies against them and this focus on mobile devices.

In [2013], Srikanth Pullela [23], discussed the performance issues like handoffs, routing etc. Then he further addressed that security is another key issue, which needs to be considered when the communication channel is set up. Also protocols are being proposed for different applications like wireless application protocol, 802.11 etc. Most of them are based on the public and private key cryptography.

In [2013], V. Gayoaso Martinez and L. Hernandez Encinas [24], have discussed the ECC, one of the best options for protecting sensitive information. The latest version of the JAVA platform includes a cryptographic provider - SunEC which implements EC operations and protocols. They have explained the complete code of three applications to generate key pairs, perform key exchanges, and produce digital signatures with EC in JAVA.

In [2013], Muhammad Waseem Khan [25], explained that short message service (SMS) is one of the frequently used mobile services with universal availability in all GSM networks but the SMS facility has not achieved secure transmission of plaintext between different mobile phone devices. However, SMS does not have its own built-in mechanism to secure the transmitted data because security is not considered as a priority application for mobile devices. The existing schemes provide room for the secure SMS message communication. The effect of each security scheme on mobile device's performance was also observed. Finally summary of all security schemes with their limitations was presented.

In [2013], Ram Ratan Ahirwal and Manoj Ahke [26], explained the Diffie-Hellman scheme as one of the key exchanging cryptosystem, and no messages are involved in this scheme and using this key and ECC for encryption and decryption. Two different methods to encrypt and decrypt the message were proposed by them. They pointed out that the second method supports the system with more security than the first method because the sender computes the exponentiation function between the coordinates of the encryption algorithm and the receiver computes the inverse of the exponentiation function between the coordinates of the key in the decryption algorithm, While in the first method, the sender compute the multiplication between the coordinates of the key in the encryption algorithm, the receiver compute the multiplication between the coordinates of the key in decryption algorithm and forward secrecy in HTTPS protocol.

In [2014], Sathish Kumar et. al. [27], have discussed about the mobile hand-held device are used in an efficient way to deliver real time data to the users in the battle field military applications and the use of security features in military applications such as data confidentiality, authentication etc., which are not readily offered by mobile environment. The energy expenditure in such an environment poses bottleneck while achieving privacy. Hence it is necessary to design and implement an energy efficient authentication protocol that accomplishes a high level of security with minimum energy consumption. They have proposed the implementation of energy efficient authentication protocol for mobile devices.

In [2014], Hamed Khiabani et. al. [28], explained the extensive deployment of wireless networking, mobile and embedded devices, other pervasive computing technologies that are prone to security threats for which nobody will be prepared for. Security and privacy are the main concerns in mobile computing which can be observed from several perspectives including hardware, operating systems, networks, databases, user interfaces, and applications.

In [2014], Seema P. Nakhate, and R.M. Goudar [29], have implemented a secured password based mutual authentication protocol for client-server computing using ECC framework which provides secure communication between client and server with the help of user email-id and mobile phone authentication device for mobile handheld device. The proposed protocol is best suited for constrained environments where the resources such as computational power, storage capacity are extremely limited. Such devices are Mobile phones, PDA's, Palmtops and Smart cards.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

In [2015], Vishnu V and Shobha R [30], discussed the security in Wireless Sensor Networks (WSN). They have applied dynamic election of Cluster Head (CH) mechanism and two evolutionary approaches SET-IBS and SET-IBOOS, since it provides security in data transmission and reduces data losses due to nodes failure, less residual energy selected in CH. It improves the lifetime of network by increasing time of FND (First Node to die).

In [2015], Tanmoy Kumar Bishoi et. al. [31], proposed an algorithm to encrypt the data using symmetric key encryption technique and now it can be improved by using variable length key.

In [2015], Sujithra M et. al. [32], due to high performance computing techniques, cryptographic algorithms are implemented and tested in Local as well as Cloud environment. They have revealed that storing mobile data in cloud increases efficiently and AES algorithm performs better when compared with other algorithms in Mean processing time but the combination of MD5+ECC+AES algorithms qualify better than Speed-Up ratio.

In [2016], Said Bouchkaren and Saiida Lazaar [33], discussed secure data transmission through Internet. They have designed and implemented a new secret key cryptosystem due to a number of iterations of encryption and decryption of data in blocks, using cellular automata and compared them with AES algorithm and also they proved that the new algorithm resists against statistical attacks, faster than AES-256, achieved good confusion and diffusion tests.

III. CONCLUSION

Mobile Computing is a new technological development due to the magnificent growth of internet community for various applications and variety of tasks that can be performed at the requirement of the users. But the requirement is that the data must be transferred in a very fast, quick and secured manner. Hence the Cryptographic tools and techniques will be more useful to achieve this. An eavesdropper or intruder can catch the information/data during the transmission. In order to prevent this, various types of cryptographic algorithms have been used. From the findings, Elliptic Curve Cryptography (ECC) is more useful and it produces more security with less number of bits compared to RSA algorithm. It has been proved that the ECC can be applied in various levels of applications and hand - held devices.

REFERENCES

1. Mavridis I., Pangalos G., "Security Issues in Mobile computing Paradigm". 1997, http://www.researchgate.net.
2. Erik Olson and Woojin Yu, "Encryption for Mobile computing", 2000.
3. Wendy Chou, "Elliptic Curve Cryptography and Its applications to Mobile Devices,2000.
4. Limor Elbaz, "Using Public Key Cryptography in Mobile Phones", White Paper,Discretix Technologies Ltd., Advanced security solutions for constrained environments, October 2002.
5. Dharma P. Agrawal et al., "Secure Mobile Computing", S.R. Das, S.K. Das (Eds.): IWDC 2003, Springer-Verlag**,** LNCS 2918, pp.265-278.
6. WHanping Lufei and Weisong Shi, "An Adaptive Encryption Protocol in Mobile Computing", Wireless/Mobile Network Security, Springer, 2006.
7. WWWWAbhishek Kumar Gupta, "Challenges of Mobile computing", Proceedings of 2nd National Conference on Challenges & Opportunities in Information Technology RIMT – IET, Mandi Gobindgarth, March 29, 2008.
8. S. Krishna Mohan Rao and Dr. A Venugopal Reddy, "Data Dissemination in Mobile Computing Environment", BIJIT – BVICAM's International Journal of Information Technology, Bharati Vidyapeeth's Institute of Computer applications and Management (BVICAM), New Delhi, Vol. 1, No. 1, January 2009.
9. M. Razvi Doomun, and KMS Soyjaudah, "Analytical Comparison of Cryptographic Techniques for Resource-Constrained Wireless Security", International Journal of Network Security, Vol.9, No.1, July 2009, pp. 82–94.
10. Jayaprakash Kar & Banshidhar Majhi, "An Efficient Password Security of Multi-Party key exchange protocol based on ECDLP", International Journal of Computer Science and Security (IJCSS), Vol.1, Issue 5, Sep. 2009.
11. Mooseeop Kim et.al., "Design of Cryptographic Hardware Architecture for Mobile Computing", Journal of Information Processing Systems, Vol. 5, No. 4, Dec. 2009.
12. Bruno P.S. Rocha et. al., "Adaptive Security protocol selection for mobile computing", Journal of Network and Computer Applications 33, 2010, pp. 569.
13. Sathish Alampalayam Kumar, "Classification and Review of Security Schemes in Mobile Computing", Wireless Sensor Network, June 2010, 2, pp.419-440.
14. Sameer Hasan Al-Bakri, Gazi Mahabubul Alam et. al., "Securing peer-to-peer mobile communications using public key cryptography: New security strategy", International Journal of the Physical Sciences Vol. 6(4), Feb. 2011, pp. 930-938.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

15. Rahat Afreen and S.C. Mehrotra, "A Review on Elliptic Curve Cryptography for Embedded Systems", International Journal of Computer Science & Information Technology Vol. 3, No 3, June 2011.
16. Helena Rifa-Pous and Jordi Herrera-Joancomarti, "Computational and Energy Costs of Cryptographic Algorithms on Handheld Devices", Future Internet 2011, 3, 31-48; doi: 10.3390/fi3010031, ISSN 1999-5903, www.mdpi.com/journal/futureinternet.
17. Jagdish Bhatta and Lok Prakash Pandey, "Performance Evaluation of RSA Variants and Elliptic Curve Cryptography on Handheld Devices", IJCSNS International Journal of Computer Science and Network Security, Vol. 11, No. 11, Nov. 2011.
18. K. Sathish Kumar et. al., "An Experimental Study on Energy Consumption of Cryptographic Algorithms for Mobile Hand-Held Devices", International Journal of Computer Applications, Vol. 40, No.1, Feb. 2012.
19. Masoud Nosrati et. al., "Mobile and Operating Systems", Computing: Principles, Devices World Applied Programming, Vol. 2, Issue 7, July 2012.
20. Ravinder Singh Mann et al., "A Comparative Evaluation of Cryptographic Algorithms", Int. J. Computer Technology & Applications, Vol 3(5), Oct. 2012, pp. 1653-1657.
21. Giripunje et al., International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, Issue 5, May 2013, pp. 704-713.
22. Ameya Nayak, "Android Mobile Platform Security and Malware Survey", IJRET: International Journal of Research in Engineering and Technology, Vol. 02 Issue 11, Nov. 2013.
23. Srikanth Pullela, "Security Issues in Mobile computing", International Journal of Research in Engineering and Technology, Vol. 02, Issue: 11, Nov. 2013.
24. V. Gayoaso Martinez and L. Hernandez Encinas, "Implementing ECC with Java Standard Edition 7", International Journal of Computer Science and Artificial Intelligence, Dec. 2013, Vol. 3 Issue. 4, pp. 134-142.
25. Muhammad Waseem Khan, "SMS Security in Mobile Devices: A Survey", Int. J. Advanced Networking and Applications, Vol. 05, Issue 2, pp. 1873 -1882.
26. Ram Ratan Ahirwal and Manoj Ahke, "Elliptic Curve Diffie-Hellman Key Exchange Algorithm for Securing Hypertext Information on Wide Area Network", International Journal of Computer Science and Information Technologies, Vol. 4(2), 2013, pp.363 – 368.
27. Sathish Kumar et. al., "An Asymmetric Authentication Protocol for Mobile Hand held Devices using ECC over Point Multiplication Method", International Journal of Advanced Research in Computer Science & Technology, Vol. 2, Jan.–March 2014.
28. Hamed Khiabani et. al., "A Review on privacy, Security and Trust issues in Mobile Computing", Collaborative outcome of University of Malaysia and MIMOS Berhad – Information Security Cluster.
29. Seema P. Nakhate and R.M. Goudar, "Secure Authentication Protocol", International Journal of Computer Networks and Communications Security, Vol. 2, No. 4, April 2014, pp. 142 – 145.
30. Vishnu V and Shobha R, "Dynamic Cluster Head (CH) Node Election and Secure Data Transaction in CWSNs", International Journal of Engineering Research, Vol. 4, Issue Special 4, May 2015.
31. Tanmoy Kumar Bishoi et. al., "An Algorithm on Text Based Security in Modern Cryptography", Journal of Computer Networking, Wireless and Mobile Communications (JCNWMC), Vol. 5, Issue 1, Jun 2015, pp.9-14.
32. Sujithra M et. al., "Mobile Data Security: A Cryptographic Approach by Outsourcing Mobile data to Cloud", Procedia Computer Science 47 (2015), pp. 480-485.
33. Said Bouchkaren and Saiida Lazaar, "A New Iterative Secret Key Cryptosystem Based on Reversible and Irreversible Cellular Automata", International Journal of Network Security, Vol. 18, No. 2, pp. 345-353, Mar 2016.

BIOGRAPHY

Dr. K. Mani is working as an Associate Professor in the Department of Computer Science, Nehru Memorial College, Puthanampatti, Tamil Nadu since 1989. After did his MCA, he got his Graduation in Operations Research from Operational Research Society of India, Kolkatta and obtained his MTech in Advanced Information Technology from Bharathidasan University, Trichy, Tamil Nadu. He has completed his Ph. D degree from Bharathidasan University relating to enhancing security and optimizing the run time in cryptographic algorithms. His current research area includes cryptography, data mining and coding theory. He has published a number of research papers in national and international journals and conferences.

Mrs. A. Mullai is working as an Associate Professor in the Department of Computer Science, Seethalakshmi Ramaswami College, Bharathidasan University, Trichy, Tamil Nadu, India since 2000. She has 15 years of experience in teaching. After did her M.Sc in Physics, MCA in Computer Applications, she got her M.Phil in Computer Science at Bharathidasan University, Trichy, Tamil Nadu. She has cleared National Level Eligibility Test (NET) conducted by University Grants Commission (UGC), New Delhi. She is currently pursuing doctor of philosophy programme at Nehru Memorial College (Autonomous), Puthanampatti and her current area of research is Cryptography in Mobile Computing. She has published research papers in national and international conferences.

Optimizing the Run Time in Mobile Devices

K. MANI[1], A. MULLAI[2]

[1]Associate Professor, Department of Computer Science, Nehru Memorial College (Autonomous), Puthanampatti, Affiliated to Bharathidasan University, Trichy, Tamil Nadu, India
email:nitishmanik@gmail.com

[2]Associate Professor, Department of Computer Science, Seethalakshmi Ramaswami College(Autonomous), Affiliated to Bharathidasan University, Trichy, Tamil Nadu, India
email: mullai_a@yahoo.com

Abstract: **Mobile computing works on the principle of broadcasting so that the information is radiated to everyone within the wave range which increases security threats and cyber attacks replicated quickly and easily. Thus, care must be taken in handling those types of attacks to provide information security while the user roams through different networks with heterogeneous security infrastructure. For that several cryptographic techniques are employed in mobile devices. Among them the public key cryptographic algorithms like RSA and ECC play a vital role in performing security. The basic building blocks of ECC is scalar point multiplication k[P] where k is a scalar and P is a point on elliptic curve. Similarly, in RSA the encryption and decryption is of the form x^e mod n where encryption /decryption key. Normally exponentiation operation takes more time than multiplication which takes more time than addition and subtraction. To reduce the time, exponentiations and multiplications are performed by repeated multiplications and additions respectively. To reduce the time further addition chain is used. In order to generate the addition chain nature inspired based algorithms like PSO and SSO are considered in this paper. Using them, the addition chains for exponent of RSA and K[P] of ECC are generated. The encryption/decryption time and energy required for encryption/decryption are also computed and the performance of the cryptographic algorithms in mobile devices are analyzed with and without the incorporation of addition chain.**

Keywords**: Decryption, ECC, Encryption, Optimization and RSA.**

I. INTRODUCTION

Mobile devices deal with heterogeneity of networks and also in ubiquitous intelligent environment with embedded computers everywhere and reliable services to the user in an easy way. Even though they have more offerings to the user, lot of challenges like disconnection, low/ high bandwidth variability, low power and resources, security risks, wide variety of devices with different capabilities and to fit more functionality into single, smaller devices. Various public key algorithms like RSA and ECC (Elliptic Curve Cryptography) are more popular to provide security in mobile devices but they may take more time for encryption and decryption. ECC gets popularity due to its shorter key length which produces same security as in RSA with larger key length. It is noted that if a cryptographic algorithm takes more time in performing operational time (where the operational time includes both encryption and decryption) which causes customer impatience and dissatisfaction. Thus, to increase the operational time in RSA and ECC, addition chain is incorporated in performing x^e mod n of RSA and K[P] of ECC where the addition chain is generated using PSO (Particle Swarm Optimization) and Simplified Swarm Optimization(SSO). Further, if the operational time gets reduced, the energy required for the same is reduced too which ultimately increases the life time of battery.

The PSO is a robust, stochastic, population-based meta-heuristic optimization algorithm that was developed by James Kennedy and Russell Eberhart in 1995. It is based on the movement and intelligence of swarms by applying the concept of social interaction with problem solving. It uses a number of agents (particles) that constitute a swarm moving around in the search space looking for the best solution. Each particle is treated as a point in a N -dimensional space which adjusts its "flying" according to its own flying experience as well as the flying experience of other particles. Each particle keeps track of its coordinates in the solution space which are associated with the best solution (fitness) that has achieved so far by that particle.

The SSO algorithm is based on the simulation of cooperative behaviour of social -spiders. In the proposed algorithm, individuals emulate a group of spiders which interact to each other based on the biological laws of the cooperative colony. The algorithm considers two different search agents (spiders): males and females. Depending on gender, each individual is conducted by a set of different evolutionary operators which mimic different cooperative behaviours that are typically found in the colony and it is compared to other well-known evolutionary methods. The comparison examines several standard benchmark functions that are commonly considered within the literature of evolutionary algorithms. The outcome shows a high performance of the proposed method for searching a global optimum with several benchmark functions.

The rest of the paper is organized as follows. Section 2 describes the various cryptographic algorithms that already exist in the literature with respect to mobile devices. A brief explanation of RSA, ECC and addition chain are discussed in section 3. The proposed addition chain based on PSO and

978-1-5090-5573-9/16 $31.00 © 2016 IEEE
DOI 10.1109/WCCCT.2016.23

SSO are presented in section 4. Experimental results are discussed in section 5. Finally, section 6 ends with conclusion.

II. RELATED WORK

Arbit and Ashwini Kumar [1], suggested Optimized ECC (O-ECC) to assist more secure and improved protocol design with easy computation mathematically. In [2],Ahmed Tariq Sadiq,discussed the Particle Swarm Optimization (PSO). The benefit of mutation in PSO (MPSO) was used as momentum and diversity tool in the population. Experimental results clearly showed that the amount of recovered key of classical ciphers and fitness function values were better than PSO.

Ahmed A.A.Esmin and Germano Lambert-Torres[3], have proposed a methodology which was used to determine the control variable settings for real power loss minimization in the transmission system employs the PSO algorithm for the optimal setting of optimal power flow (OPF) based on loss minimization (LM) function has examined, tested on IEEE 14, 30, 118 Bus systems and the results were compared.

In[4],G.Prakash and Dr.M.Kannan, discussed that the cryptographic smart cards were used for most of the online transactions. They designed a secure technique by integrating both cryptography and steganography which could be used for smart card security. Initially, user's confidential details were encrypted using the most secure ECC technique and then the encrypted cipher was embedded into the users 'photographic image using steganography named Optimized Modified Matrix Encoding (OMME) algorithm.

Cuevas, E. Cienfuegos et al. [5], proposed the swarm intelligence models with collective behaviour in swarms of insects or animals called the social spider optimization for solving optimization tasks. The outcome showed a high performance for searching a global optimum with several benchmark functions. In [6], Wilayat Khan et al. discussed the mobility which was one of the major features of wireless communication systems and handheld devices form a major part of the systems. The limited resources like battery, memory, and computational power of these devices was a bottle neck in the security of such devices were also discussed.

Rangit j. Bhosale et al. [7], proposed the mobile ad-hoc networks (MANET) in wireless technology, having features like dynamic topology and self-configuring ability of nodes. They surveyed Intrusion detection system (IDS) that was one of the most active fields of research in MANET. Swapna B. Sasi and N. Sivanandam[8]analyzed the performance of the different methods and compared with various parameters such as maximum number of keys stored, battery capacity, runtime. They also concluded that high storage and energy was required for storing the keys.

In[9], Dolly U. Jeswani et al., discussed the cryptographic algorithms which were the key factor of the security mechanisms used for data storage and uninterrupted network transmissions. To identify the security risk associated with AES algorithm, a computational intelligence based approach for known cryptanalysis was used. A PSO oriented cryptanalysis technique for breaking the key used in Advance Encryption Standard (AES) algorithm was also introduced. The key used in AES was detected effectively with PSO. Swarm Intelligence based Cryptanalysis provided a best and optimized solution.

In [10], Chia-Ling Huang and Wei-Chang Yeh, demonstrated to optimize the reliability redundancy allocation problems (RRAP) for the series-parallel system, the complex (bridge) system, and the over speed protection of gas turbine system. Their objective of the RRAP was to maximize the system reliability for numerous decades. For that number of redundant components and the reliability of corresponding components in each subsystem with nonlinear constraints was considered simultaneously but it was more difficult for the RRAP. Hence, the RRAP was the mixed-integer programming problem with the nonlinear constraints that belongs to the NP-hard problem. To solve the RRAP, SSO algorithm was proposed to improve the computation efficiency and found that it outperforms the previously best-known solutions.

In [11], Ji Weidong and Zhu Songyu discussed and PSO which was one of the most common algorithms for optimization because of its simple, convenient and good robustness. They have proposed a new particle swarm algorithm as improved cut PSO algorithm based on filtering mechanism (ELPSO) to improve its operation speed and more accurate.

In [12], Jin Yang et. al. discussed that the Mobile sinks that could achieve load -balancing and energy-consumption balancing across the wireless sensor networks (WSNs). However, the frequent change of the paths between source nodes and the sinks caused by sink mobility introduces significant overhead in terms of energy and packet delays. They have presented the enhanced version of the network performance of WSNs with mobile sinks (MWSNs), in an efficient routing strategy using PSO to build the optimal routing paths. A novel greedy discrete particle swarm optimization with memory (GMDPSO) was introduced to improve the greedy forwarding routing, a greedy search strategy was designed to drive particles to find a better position quickly, searching history was memorized to accelerate convergence. Simulation results demonstrated that the new protocol significantly improved the robustness and adapted to rapid topological changes with multiple mobile sinks, while efficiently reducing the communication overhead and the energy consumption.

III. CONCEPT OF ECC, RSA AND ADDITION CHAIN

This section includes the definition of EC, point addition on EC, k[P], working principle of RSA and addition chain.

A. EC *Definition Over GF(P):*

Let p be a prime greater than 3 and a and b be two integers such that $4a^3+27b^2 \neq 0 \pmod p$. EC over the finite field F_p is the set of points (x, y) ϵ $F_p \times F_p$ satisfying the Weiestrass equation

$$E: y^2 \equiv x^3+ax+b \qquad \ldots(1)$$

together with point at infinity O. The point of infinity is similar to the number 0 as in normal addition. Thus, for all points in EC, P + O = P and P+(-P)= 0 [14]. An abelian group is formed by the point on E with the addition operation. –P= $(x_1,-y_1)$ is the inverse of the point P= (x_1, y_1).

Point Addition on EC

Let P= (x_1, y_1) Q=(x_2, y_2) be points in EC, given in affine coordinates. Assume P, Q $\neq$ O and P $\neq$ -Q. The sum R=(x_3, y_3) = P+Q is computed as

if P≠Q, i.e., point addition,

$$\lambda = (y_2-y_1) \div (x_2-x_1) \qquad \ldots(2)$$

$$x_3= \lambda^2-x_1-x_2,\ y_3 = (x_1-x_3)\lambda-y_1$$

The cost required for computing point addition is (1I+3M)

If P=Q, i.e., point doubling,

$$\lambda = (3x_1^2-y1) \div 2y_1\ , \qquad \ldots(3)$$

$$x_3= \lambda^2-2x_1,\ y_3 = \lambda\ (x_1-x_3)\ -y_1$$

The cost required for computing point doubling is (1I+4M)

Scalar Point Multiplication

The k[P] is similar to the exponentiation operation in other public- key cryptosystems like RSA. It is noted that exponentiation operation normally takes more time than multiplication and hence the exponentiation operation may be performed as repeated multiplication. Similarly, in ECC, k[P] is performed by adding p to itself k times [14]. Thus,, for integer the computation k[P] is given by

$$k[P] = \begin{cases} \underbrace{P+P+\ldots+P}_{k \text{ times}} & \text{for } k>0 \\ 0 & \text{for } k=0 \\ [-k]-P & \text{for } k<0 \end{cases} \qquad \ldots(4)$$

It is one of the public-key cryptography based on the algebraic structure of elliptic curves over finite fields and also it gets popularity due to its shorter key length. Further it reduces space for key storage, arithmetic cost, and time during the transmission of keys.

B. RSA

RSA was proposed by Rivest et.al. The private key of a user consists of two prime p and q and an exponent (decryption key) d. The public-key consists of the modulus n = pq, and an exponent e such that $d = e^{-1} \bmod (p-1) \times (q-1)$. To encrypt a plaintext M the user computes $C = M^e \bmod n$ and decryption is done by calculating $M = C^d \bmod n$.

C. Addition Chain

An addition chain is a finite sequence of positive integers called elements, $1= a_0 \leq a_1 \leq a_2 \leq \ldots \leq a_r = e$ with the property that for all $i>0$ there exist a_j, k with $a_i=a_j+a_k$ and $r \geq i \geq j \geq k \geq 0$. This is called an addition chain of length r for the target e. An optimal addition chain is the one which has the shortest possible length denoted by $l(e)$and it is a strictly increasing sequence as duplicate chain elements could be removed to shorten the chain. It is noted that for the given integer e, more number of addition chains are possible. But for finding at least one of the shortest addition chain is an NP-hard problem. For example, n=170. All possible optimum addition chains are
1-2-3-5-10-20-40-45-85-170 1-2-3-5-10-20-40-80-85-170
1-2-3-5-10-20-40-80-90-170 1-2-3-5-10-20-40-80-160-170
1-2-4-5-10-20-40-45-85-170 1-2-4-5-10-20-40-80-85-170
1-2-4-5-10-20-40-80-90-170 1-2-4-5-10-20-40-80-160170 1-2-4-6-10-20-40-80-90-170 1-2-4-6-10-20-40-80-160-170

IV. PROPOSED METHODOLOGY

Most of the public-key cryptosystems like RSA, ElGamal, etc., modular exponentiation is the cornerstone operation which plays a vital role in performing encryption/decryption operations. They often involve raising large elements of some group fields to large powers. Successive multiplication is normally used to perform modular multiplication but it is a time-consuming process. For example, to compute x^e based on paper-and pencil method, it requires (e-1) multiplication of x. i.e., $x^1\ x^2\ x^3\ \ldots\ x^{e-1}\ x^e$. Similarly, in ECC to perform encryption/ decryption, scalar point multiplication k[P]mod m, where P is a elliptic curve point, k is an arbitrary integer in the range 1< k <ord(p), and m is a modulus plays a vital role. To reduce the number of multiplications further in public key cryptography like RSA and number of additions in k[P] of ECC addition chain is used.

To generate the addition chain for an integer using PSO, the particle represents the addition chain and N represents the number of addition chains (particle's population). Before finding the optimal addition chain, the addition chain for *e* is tentatively taken and its corresponding length *l(e)* is termed as pbest. Thus, the i^{th} particle represents i^{th} addition chain and it is represented as $Xi=(a_{i1}, a_{i2}, \ldots, a_{in})$, i=1,2,…,N. It is noted that for the given integer *e,N* number of addition chains are generated and the optimal addition chains are always the subset of N. Also,pbest (global best particle) represents the optimal addition chain. The velocity of the particle is the next number to be selected in the addition chain called intermediate numbers a_{ij},where $3 \leq j \leq l-1$, where *l* is the last number in the addition chain. This is because in any addition chain the first number $a_{i1}=1$; $a_{i2}=2$ and a_{in}= e. These numbers are generated either from addition or doubling steps from the previous numbers occur in the addition chain.

In order to generate the intermediate numbers, the random numbers *rand1* and *rand2*are used to select the addition or doubling steps respectively and they indicate the maximum range of uniform random number. Suppose the selected random number is <*rand1*, to generate the next number addition step is considered otherwise doubling step is considered. It is noted that more than one numbers are generated using addition step, to select the next number from the current number the maximum range specified in *rand1* is divided into number of numbers generated using addition step. For example, if the current number is *5*, from *5* the numbers *6, 7, 8, 9* are generated using addition step and the range of random number is selected as *rand1/4* . Suppose *rand1=0.5*, then the probability of selecting each number is *5/4=.125*. Thus, if the uniform random number *U is in the range0.000≤U< 0.125*, then the next number selected in the addition chain is *6* from *5*, if*0.125≤U<0.250*, then the next number selected is *7*. Similarly, if *0.250≤U<0.375*, then *8* is selected and if *0.375≤U<0.5* Once the next number of addition chain is generated the process is recursively performed till it reaches *e* and its l(e) is found called as new *l(e)*. Then the new*l(e)* is compared with *pbes*t. If it is <*pbes*t, now *pbest*= new *l(e)*. Otherwise, previous value of *pbes*t is retained. Similarly, for all the numbers obtained from current numbers, the said process is repeated till the optimal addition chain and its corresponding length is found. The concept used in SSO is also used for generating the optimal addition chain for an integer e.

A. Proposed Methodology- RSA-PSO Based Addition Chain-An Example

In order to get a proper understanding of the subject matter of this paper using RSA, let p=13, q=17 and e=11, then n= 13(17)=221, (p-1)(q-1)=12(16)=192. Now d = 133. To encrypt, $C = M^{11} \bmod 187$ and to decrypt $M = C^{133} \bmod 187$. As e=11, the conventional repeated multiplication requires 10 multiplications. But, if the PSO based addition chain is used, the addition chain for *e=11* is 1-2-3-5-10-11; *l(e)*=5 which requires only 5 multiplications. Similarly, for *d=133*, conventional process requires *132*multiplications whereas it requires only *10* multiplications if PSO based addition chain is used because the addition chain for *d=133*is 1-2-3-5-10-20-30-50-100-130-133 and *l(d)=10.*

B. Proposed Methodology- ECC-PSO Based Addition Chain-An Example

To encrypt the message using ECC, Diffie-Hellman key exchange protocol is used in this paper. For that let p=211; and Ep(0, -4) and G=(2,2). A's private key is n_A=121, so A's public key is P_A=121(2,2)=(115,48). B's private key is n_B=203, so B's public key is P_B= 203(2,2)=(130,203), The shared secret key is 121(130,203)=203(115,48)=(161, 169). It is noted that in P_A, k=121. If the conventional repeated addition chain is used, it requires 120 additions whereas ECC-PSO based addition chain, it requires only 10 additions because, the addition chain for k=121 is 1-2-3-5-10-20-40-50-100-120 -121 and *l(k)=10.* Similarly, for P_B, k=203, which requires only 10 multiplications because its addition chain is 1-2-3-5-10-20-40 -50 -100-200-203; *l(k)=10.*which is far less than202 number of addition if the conventional repeated addition chain is used.

V. EXPERIMENTAL RESULTS

The proposed methodology is implemented in VC++ with android emulator for varying file sizes using RSA and ECC. Table 1 and Table 2 show the encryption/decryption time and encryption/decryption power using RSA and ECC without addition chain respectively and also their corresponding graphical representations of encryption/decryption time and energy required for encryption and decryption operations are shown in Fig. 1, Fig. 2, Fig. 3 and Fig. 4 respectively.

Table 1: Operational Time and Power Consumption using RSA without Addition Chain

FS(in MB)	Time (in Ms)		Power (in mW)	
	E	D	E	D
1	1664	1643	571	570
2	3233	3190	1096	1090
4	6492	6484	2177	2181
8:	13672	13666	4569	4569
16	27417	27420	9163	9148

FS- File Size E-Encryption Time D-Decryption Time

Table 2: Operational Time and Power Consumption using ECC without Addition Chain

FS(in MB)	Time (in Ms)		Power (in mW)	
	E	D	E	D
1	2419	2313	848	777
2	4788	4517	1596	1506
4	9562	9225	3192	3075
8:	20182	19377	6734	6466
16	40434	38939	13504	13000

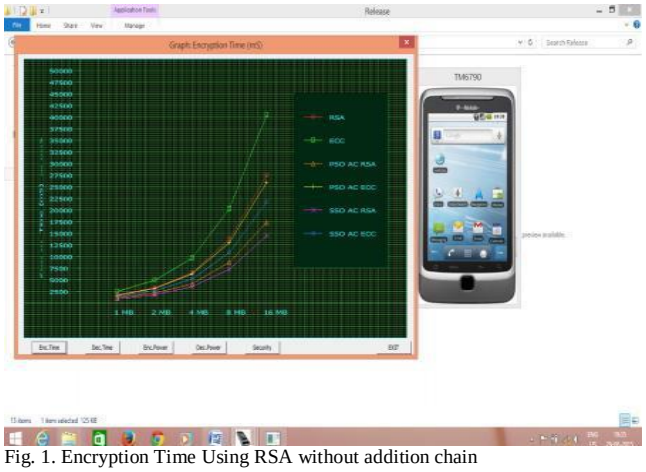


Fig. 1. Encryption Time Using RSA without addition chain

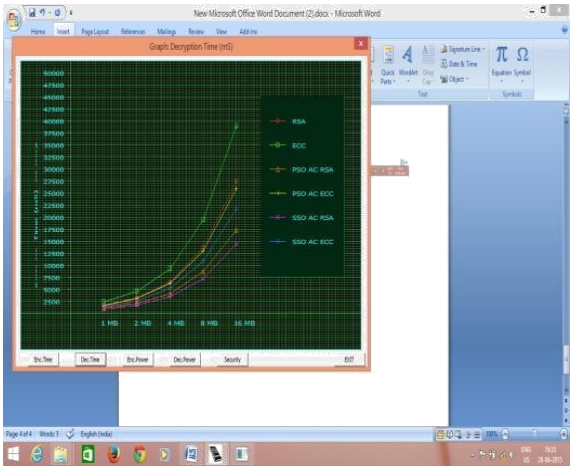


Fig. 2. Decryption Time Using RSA without addition chain

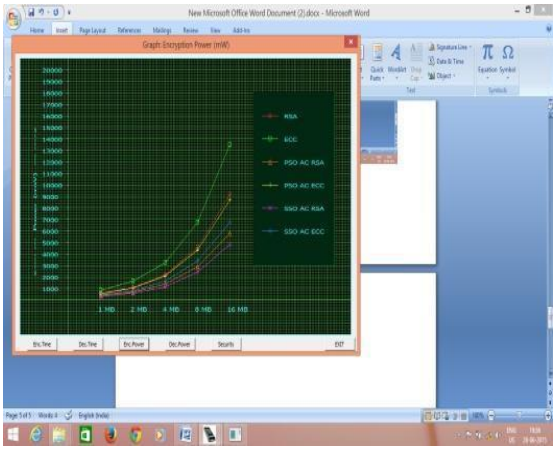


Fig. 3 Encryption Power TimeUsing ECC without addition chain

Fig. 4. Decryption Time Using ECC without addition chain

After incorporating PSO addition chain in RSA and ECC, the time taken for encryption/decryption operation and the power consumption for the same operations are shown in Table 3 and Table 4 respectively.

Table.3:Operational Time and Power Consumption using RSA with PSO Addition Chain

FS(in MB)	Time (in Ms)		Power(in MW)	
	E	D	E	D
1	1034	1022	361	349
2	2036	2010	679	669
4	4102	4100	1384	1378
8	8654	8619	2902	2888
16	17326	17311	5792	5781

Table. 4: Operational Time and Power Consumption using ECC with PSO Addition Chain

FS(in MB)	Time(in mS)		Power(mW)	
	E	D	E	D
1	1580	1555	541	536
2	3079	3013	1036	1013
4	6148	6143	2052	2050
8	12979	12920	4350	4309
16	25997	25968	8686	8656

Table. 5:Operational Time and Power Consumption using RSA with SSO Addition Chain

FS(in MB)	Time(mS)		Power(mW)	
	E	D	E	D
1	869	864	294	291
2	1696	1680	575	564
4	3420	3416	1144	1147
8	7206	7182	2416	2397
16	14442	14424	4827	4809

After incorporating SSO addition chain in RSA and ECC, the time taken for encryption/decryption operation and the power consumption for the same operations are shown in Table 5 and Table 6 respectively.

Table. 6:Operational Time and Power Consumption using ECC with SSO Addition Chain

FS(inMB)	Time(mS)		Power(mW)	
	E	D	E	D
1	1291	1291	408	417
2	2560	2515	813	785
4	5119	5118	1593	1604
8	10818	10789	3367	3369
16	21664	21633	6739	6731

From Table 1 and Table 2, it is observed that ECC takes more time than RSA for both operational and power consumption. This is because lot of computations like generation of points on EC, addition of points in performing k[P] are involved in ECC when it is compared with RSA. It is evident from Table 3 and Table 4 that the time required for operational and power consumption time is substantially reduced using RSA and ECC with PSO based addition chain when the same is compared without addition chain. It is observed from table 5 and table 6, same is also happening in the case of RSA and ECC with SSO based addition chain. But SSO based addition chain always takes less time when it is compared with PSO if they are incorporated in RSA and ECC. This is because in PSO based addition chain, computations of velocity and inertia weight take more time than SSO because based on random number alone the next number in the addition chain is determined.

VI. CONCLUSION

PSO and SSO based addition chain for the integers are thought of and they are incorporated in public key cryptographic algorithms like RSA and ECC which play a vital role in optimizing the runtime in mobile devices. To implement the proposed algorithms Android emulator has been chosen for the experimental setup. The proposed algorithms significantly reducing the encryption and decryption time in both RSA and ECC because the said algorithms minimize the number of multiplications and additions in exponentiation and k[P] respectively which will eventually result in minimizing the energy required for both encryption and decryption.

REFERENCES

[1]. Arbit and Ashwini Kumar, “Optimized Elliptic Curve Cryptography as Fine Balance for Wireless Sensor Network”, International Journal of Modeling and Optimization, Vol.1, No. 4, October 2011.

[2]. Ahmed Tariq Sadiq, “Mutation-Based Particle Swarm Optimization (MPSO) to Attack Classical Cryptography Methods”, Journal of Computer Science and Technology Research 2 (2012) 50-65, ISSN:2231-8852, March 2012.

[3]. Ahemed A. A.Esmin and Germano Lambert-Torres, "Application of Particle Swarm Optimization to optimal power systems", International Journal of Innovative Computing, Information and Control, Vol. 8, No. 3(A), pp. 1705-1716, March 2012.

[4]. G.Prakash and Dr.M.Kannan, “Enhancing Security in Cryptographic in Smart Cards through Elliptic Curve Cryptography and Optimized Modified Matrix Encoding Algorithms”, Journal of Theoretical and Applied Information Technology, Vol. 58, No.3, December 2013.

[5]. Cuevas, E., Cienfuegos, M., Zaldívar, D., Pérez-Cisneros, M. A swarm optimization algorithm inspired in the behaviour of the social-spider, Expert Systems with Applications, Vol. 40, No. 16, 2013.

[6]. Wilayat Khan, Habib Ullah and Riaz Hussain, “Energy Efficient Mutual Authentication Protocol for Handheld devices based on Public Key Cryptography”, International Journal of Computer Theory and Engineering, Vol. 5, No. 5, October 2013.

[7]. Rangit j. Bhosale et al, “A Survey on Intrusion detection System for Mobile Ad-hoc Networks”, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5, No. 6, 2014.

[8]. Swapna B. Sasi and N. Sivanandam, “A Survey on Cryptography using Optimization algorithms in WSNs”, Indian Journal of Science and Technology, Vol. 8. No. 3, February 2015.

[9]. Dolly U. Jeswani and Swati G. Kale, “The Particle Swarm Optimization Based Linear Cryptanalysis of Advanced Encryption Standard Algorithm”, International Journal on Recent and Innovation Trends in Computing and Communication, Vol. 3, April 2015.

[10]. Chia-Ling Huang and Wei-Chang Yeh," Simplified Swarm Optimization Algorithm for reliability redundancy allocation problems", IEEE Computer Society, 2015.

[11]. Ji Weidong and Zhu Songyu, "A Filtering Mechanism Based Optimization for Particle Swarm", International Journal of u- and e-Service, Science and Technology Vol.9, No. 1, 2016.

[12]. Jin Yang, Fagui Liu, Jianneng Cao and Liangming Wang, "Discrete Particle Swarm Optimization Routing Protocol for Wireless Sensor Networks with Multiple Mobile Sinks", MDPI journals , Sensors 2016.

Enhancing the security in RSA and elliptic curve cryptography based on addition chain using simplified Swarm Optimization and Particle Swarm Optimization for mobile devices

A. Mullai & K. Mani

International Journal of Information Technology
An Official Journal of Bharati Vidyapeeth's Institute of Computer Applications and Management

ISSN 2511-2104
Volume 13
Number 2

Int. j. inf. tecnol. (2021) 13:551-564
DOI 10.1007/s41870-019-00413-8

Springer

Int. j. inf. tecnol. (April 2021) 13(2):551–564
https://doi.org/10.1007/s41870-019-00413-8

ORIGINAL RESEARCH

Enhancing the security in RSA and elliptic curve cryptography based on addition chain using simplified Swarm Optimization and Particle Swarm Optimization for mobile devices

A. Mullai[1] · **K. Mani**[2]

Received: 15 November 2018 / Accepted: 16 December 2019 / Published online: 7 January 2020

Abstract Security is the major concern in mobile or portable devices because the internet community can do their work at any time at any place at anywhere. Today various cryptographic algorithms like RSA, Elliptic Curve Cryptography (ECC), etc., can be used to protect the information in mobile devices. But, they have some limitations viz., energy, battery power, processing speed, operating systems, screen size, resolution, memory size, etc. Providing security for limited power mobile devices is a challenging task. RSA and ECC are normally used in mobile devices. In RSA, both encryption and decryption are of the form x^e mod n and in ECC, the scalar point k[P] where k is a scalar and P is a point in EC plays a vital role in performing encryption and decryption. The point arithmetic involved in ECC is a power starving process. To speed up the operations in both cryptographic algorithms, addition chains (AC) are normally used. If the encryption and decryption time get reduced, it ultimately reduces the power consumption. There are several AC algorithms exist in the literature. But, ACs are generated using Particle Swarm Optimization and Simplified Swarm Optimization are proposed in this paper and they are used in the said processes of RSA and ECC with two android and window emulators. The processing time, power consumption taken for encryption, decryption process and security of the said algorithms are also analysed.

Keywords RSA · ECC · Addition chain · PSO and SSO

1 Introduction

Mobile devices are primarily battery powered. The power has to be utilized optimally to improve the mobility and life time of the mobile node. Data involved in today's mobile communications are diverged in sensitivity from insensitive public social media data to highly confidential delicate private data. Ensuring security in present mobile network is a critical task because high security protocols have to be used and the power consumption of the security architecture is to be kept in control. Current security algorithms in practice are having a general nature in common wherever security is improved and their power utilization may be high. Computing and updating of security keys are taking more power consumption which are directly proportional to the size of the security keys. Larger size keys provide higher security and they consume more computational power as well. There are many famed cryptography procedures are used for mobile security. RSA [1] and ECC are used for digital data security in a great extend. RSA is a procedure of computational simplicity whereas ECC provides greater security. Large prime numbers are used as security keys in these methods. At present while comparing many cryptography methods, ECC [2, 3] provides more security even with lesser key sizes. In ECC, k[P] [4, 5] plays a vital role in performing encryption and decryption process. Similarly in RSA, the encryption and decryption are of the form x^e mod n. Since the exponentiation

✉ A. Mullai
mullai_a@yahoo.com

K. Mani
nitishmanik@gmail.com

[1] Department of Computer Science, Seethalakshmi Ramaswami College (Autonomous), Affiliated to Bharathidasan University, Trichy, Tamil Nadu, India

[2] Department of Computer Science, Nehru Memorial College (Autonomous), Affiliated to Bharathidasan University, Puthanampatti, Trichy, Tamil Nadu, India

operation takes more time than any other arithmetic operations, to minimize the time, exponentiations are performed by repeated multiplication. Similarly, in the case of k[P] of ECC, the multiplications should be performed by repeated addition. To reduce the number of multiplications in RSA and number of additions in k[P] further, ACs are used. The k[P] calculations involved in ECC makes it perplexed to use in limited power mobile nodes.

AC [6] is a sequence of integers in which each succeeding element should be the sum of two preceding elements or the doubled value of a preceding element. In general, ACs are started with the positive integers 1 and 2 where the third element may be 1 + 2 = 3 or 2 + 2 = 4. Most of the public-key cryptography procedures operate in the basis of computing of modular exponentiations.

The construction [7, 8] of each element of an AC is called a step. For an AC,

$$1 = a_0 \leq a_1 \leq \cdots \leq a_r = n. \tag{1}$$

The following steps are involved.

Doubling step:

$$a_i = 2a_{i-1}, \quad i > 0. \tag{2}$$

Non-doubling step:

$$a_i = a_j + a_k, \quad i > j > k \geq 0. \tag{3}$$

The steps of the form $a_i = 2a_{,j}\ j \leq i - 2$ are defined as non-doubling steps.

Big step:

$$\lambda(a_i) = \lambda(a_{i-1}) + 1. \tag{4}$$

Small step:

$$\lambda(a_i) = \lambda(a_{i-1}). \tag{5}$$

Finding minimal length ACs helps in diluting the complexity of modular exponentiations. Minimal length ACs [9, 10] can be calculated either using brute force method or some clever algorithm. Brute force method is not recommended because of its high computational cost. A few intelligent procedures are in use to calculate minimal length ACs.

There is no defined procedure available to produce minimal length AC for all numbers. These methods are consuming different computational powers to calculate minimal length ACs. A new procedure is presented in this paper to generate it using Swarm Optimizations [11] like Particle Swarm Optimization (PSO) [8, 12] and Simplified Swarm Optimization (SSO) [13]. Generated ACs by both methods are used in RSA and ECC to measure the standard crypt-analyse parameters like encryption time, decryption time, power consumption and security levels. They are termed as RSA-Particle Swarm Optimization based AC (RSA-PSOAC), ECC-Particle Swarm Optimization based AC (ECC-PSOAC) and RSA-Simplified Swarm Optimization based AC (RSA-SSOAC) and ECC-Simplified Swarm Optimization based AC (ECC-SSOAC). Files with different sizes are encrypted and decrypted in android and windows mobile operating environments to measure the said metric parameters.

The rest of the paper is organized as follows: Sect. 2 describes the concepts involved in RSA, ECC, PSO and SSO. Existing works are given in Sect. 3. The proposed methodology for generating AC using PSO and SSO are discussed in Sect. 4. The experimental set up is shown in Sect. 5. The results obtained after incorporating AC into RSA and ECC encryptions in two different OS are discussed in Sect. 6. Finally, Sect. 7 ends with conclusion with future extension of this work.

2 Concepts of RSA, ECC, PSO and SSO

This section explains the concepts involved in RSA, ECC, PSO and SSO.

2.1 RSA

RSA is a public key cryptography and it was proposed by Rivest et al. [6, 14]. The public-key of a user consists of two prime p and q and an exponent (encryption key) e. The private-key consists of the modulus n = pq, and an exponent d such that $d = e^{-1} \bmod (p-1)(q-1)$. To encrypt a plaintext M, the user computes $C = M^e \bmod n$ and decryption is done by calculating $M = C^d \bmod n$. This asymmetric crypto-procedure is widely used in the digital age to secure delicate data because of its simplicity.

2.2 ECC

ECC was proposed in 1985 by Neal Koblitz and Victor Miller [15–17]. It is an alternative to the established cryptosystems like RSA, ElGamal, Rabin, etc. It guarantees all the security services with the shorter keys. The use of shorter length implies less space for key storage, less arithmetic cost and time saving when keys are transmitted. ECC produces same level of security with 160-bit keys as other methods security with 1024-bit keys.

2.2.1 Definition (elliptic curve)

Let p be a prime greater than 3 and a and b be two integers such that $4a^3 + 27b^2 \neq 0 \pmod p$.

EC over the finite field F_p is the set of points $(x, y) \in F_p \times F_p$ satisfying the Weiestrass equation

$$E : y^2 \equiv x^3 + ax + b \tag{6}$$

together with point at infinity O. The point of infinity [18] is similar to the number 0 as in normal addition.

2.2.2 Point addition and doubling on EC

Let P = (x_1, y_1) and Q = (x_2, y_2) be two points in EC, given in affine coordinates. Assume P, Q ≠ 0 and P ≠ − Q. The sum R = (x_3, y_3) = P + Q is computed as if P ≠ Q, i.e., point addition,

$$\lambda = (y_2 - y_1)/(x_2 - x_1), \quad x_3 = \lambda^2 - x_1 - x_2, \\ y_3 = (x_1 - x_3)\lambda - y_1. \tag{7}$$

If P = Q, i.e., point doubling,

$$\lambda = \left(3x_1^2 - y_1\right)/2y_1, \; x_3 = \lambda^2 - 2x_1, \\ y_3 = \lambda(x_1 - x_3). \tag{8}$$

In ECC, k[P] is performed by adding p to itself k times [19]. Thus, for integer the computation

$$k[P] \text{ is given by } k[P] = \underbrace{P + P + \cdots + P}_{\text{k times}}, \quad for\ k > 0. \tag{9}$$

2.3 Particle Swarm Optimization (PSO)

PSO was developed in 1995 by Dr. Eberhart and Dr. Kennedy based on the social behaviour of flocking birds and fish schooling [20, 21]. It is initialized with a population of multiple random solutions. The results are optimized through iterations towards the best result and it acquires an optimum solution by tiding generations. While a group of birds searching for food in an area, their initial locations are random. The birds do not know the place of the food initially. But they get closer to the food after a set of movements that is iterations. The best way to reach food is following the way of the bird which is nearest to food. All particles are updated after each iteration based on two *best* values. The first *best* value *pBest* is achieved by a particle as yet. Second *best* value *gBest* is the best value achieved by the overall population towards the fitness function trailed by the particle swarm optimizer [22]. There are two equations defined to determine the velocity and the position of the particles.

Velocity equation:

$$\forall \in \{1,\ n\} : v_i \\ = v_i + c_1 + y_1 x(pb_i - p_i) + c_2 \times y_2 \times (Gb_i - P_i), \tag{10}$$

where, n is the number of maximum permitted iterations, v is the velocity of the particle, p_b is the *pbest* (particle best), p is the present position, G is the *gbest* (global best), c_1, c_2 is the learning factors.

In general c_1 = c_2 selected from the range of 0–4 y_1, y_2: Random numbers between 0 and 1,

$$\text{Position Equation}: \quad P = P + V. \tag{11}$$

The steps involved in PSO algorithm are:

1. Initialize all particles
2. Calculate fitness value for each particle
3. If the calculated fitness value *pBest* is better than existing *pBest,* then update *pBest*
4. Find the particle with best fitness value *gBest* from overall population
5. For all particles, calculate velocity and position based on the equations and update values
6. Repeat from Step ii until maximum number of iterations achieved or optimum result achieved.

2.4 Simplified Swarm Optimization (SSO)

SSO was proposed by Wei-Chang Yeh. It is an evolutionary computational swarm based intelligence method and it is used in many recent research fields because of its efficiency and flexibility. SSO procedure is commenced with initial population of particles embedded with a pair or finite-length encoded string and a fitness value. In this process, each individual particle refers a solution. An updating mechanism (UM) [23] is defined to improve the solutions through iterations. The UM of SSO is declared as,

$$X_{ij}^t = \begin{cases} X_{ij}^{t\cdot 1} & if\ p \in [0,\ C_w] \\ P_{ij}^{t\cdot 1} & if\ p \in [C_w,\ C_p] \\ g_i & if\ p \in [C_p,\ C_g] \\ x & if\ p \in [C_g,\ 1] \end{cases} \tag{12}$$

where X_{ij}^t refers the position of ith particle with respect to jth variable of the solution space at tth generation.

$$p_i = (P_{i1},\ P_{i2}, \ldots P_{id}) \tag{13}$$

where *d* refers total number of variables in the problem. P_i is otherwise referred as particle best value. *pBest* is the best solution with best fitness value of its own history.

The overall best solution of entire population is represented as global best value *gBest* which is referred as *g*,

$$g = (g_1, g_2, \ldots g_d) \tag{14}$$

where g_i refers jth variable in *gBest*, x is a random value between the lower bound and upper bound of jth variable, ρ is a consistent random number between 0 and 1, c_w, c_p and c_g are predetermined parameters.

The steps involved in SSO are:

1. Initialize $X_{0,i}$ = P_i randomly
2. Let η = Number of maximum solutions
3. Let t = 1 and i = 1, 2,…η

4. Calculate $F(X_{0,i})$ and find *gBest*
5. Initialize i = 1
6. Calculate $X_{t,i}$ from $X_{t-1,i}$ based on UM
7. If F ($X_{t,i}$) is better than F (P_i), then $P_i = X_{t,i}$, else go to Step ix
8. If F (P_i) is better than $F(P_{gBest})$, then *gBest* = i
9. If $i < \eta$ then let $i = i + 1$ and go to step iv.

SSO differs from PSO by its updating mechanism UM which is used to maintain population diversity and to reduce the local optimum lockouts.

3 Related works

The major challenges in mobile computing are low bandwidth, high error rate, power restrictions, security, limited capabilities, disconnection and the problems created due to the mobility of the user. Applying cryptographic algorithms to the mobile units would not be accessed by the hackers. The existing works in this area are listed as follows.

Mavridis I and Pangalos G, discussed the operational and security issues of mobile components in distributed environments in wireless networking using the mobile agents and applied some security mechanism in a healthcare paradigm [24]. Erik Olson and Woojin Yu, surveyed various symmetric key algorithms and their usage in mobile computing, and proposed the architecture used in the processor is similar to 68K processor [25]. Wendy Chou, surveyed the explosive growth in the usage of mobile and wireless devices demands a new generation of public key cryptography (PKC) schemes, their limitations due to power, bandwidth for providing security in mobile devices [26].

Limor Elbaz, implemented PKC in security of wireless devices and the use of Public Key Infrastructure (PKI) in current and future applications of mobile phones [27]. Dharma P. Agrawal et al., analyzed some security issues, various threats in the existing countermeasures and concluded that encryption plays an important role for secured communication in mobile computing environments [28]. Hanping Lufei and Weisong Shi, proposed an adaptive encryption protocol to choose a proper encryption algorithm dynamically which enhances security and minimizes the time overhead [29]. Abhishek Kumar Gupta, explained the field of mobile computing (computing and communication) dependent on information, it is available only by accessing a network and discussed that the mobility can also cause wireless connections to be lost or degraded [30].

S. Krishna Mohan Rao and A. Venugopal Reddy, discussed the data dissemination which illustrates quickly access of the data item in mobile devices with minimum access time so that the mobile clients save the precious battery power while they are moving from one place to another [31]. The resource-limited wireless sensor networks (WSN) needs reliable and efficient security mechanisms using two block ciphers, namely the RC5 and AES, the suitability of the algorithm for resource-limited wireless network security is analyzed by M. Razvi Doomun, and K.M.S. Soyjaudah [32].

Kar and Banshidhar Majhi proposed an efficient password security of multi-party key exchange protocol based on elliptic curve discrete logarithm problem (ECDLP), when communicating over a public network to establish a common secret key called session key and also build protocol for password authentication model for group members to hold an individual password rather than a common password and two one-way hash functions to build the security [33].

Mooseeop Kim et al., proposed a compact architecture for a cryptographic engine on a mobile platform, is highly effective to implement the scalable RSA and unified SHA algorithms with a minimum resource usage. The combined performance results of circuit area, power efficiency, throughput, and functionality is suitable for mobile computing systems [34]. Bruno P.S. Rocha et al., demonstrated a security service, which works as a middleware, to dynamically change the security protocols used between two peers and provide the solution for performance gain in the execution of cryptographic primitives [35].

Sathish Alampalayam Kumar, suggested the security solutions for various types of attacks proposed by various schemes and the open research issues in providing security for mobile agent based computing systems [36]. Sameer Hasan et al., proposed a non-server architecture PKC to secure the mobile communications and implemented various security services needed for mobile communication. Compared with server based architecture, this has low risk and the security has been improved. They used NTRU algorithm for public key cryptography in non-server architecture [37].

Rahat Afreen and S.C. Mehrotra, discussed the ECC implementation in hardware as well as software platforms analysis and Helena Rifa-Pous and Jordi Herrera-Joancomarti discussed the performances of different cryptographic algorithms in PDAs and compared in terms of device cost, operating system, screen, and network interfaces to determine the overhead and the results were used to estimate the costs of network security protocols design [38, 39].

Jagdish Bhatta and Lok Prakash Pandey, proposed a software level cryptographic protocol to measure the energy level through the device's serial port and their power consumption which provides better security and acquires very less consumption of energy than the existing

cryptographic protocols [40]. K. Sathish Kumar et al., designed and implemented an energy efficient authentication protocol that accomplishes a high level security with minimum energy consumption for mobile devices [41].

Masoud Nosrati et al., proposed an algorithm for security mechanism in different types of mobile devices use some algorithms to scramble data into unreadable text which can be only decoded or decrypted by those who possess the associated key and these algorithms consume a significant amount of computing resources such as CPU time, memory, battery power and computation time [42].

Ravinder Singh Mann et al., presented the comparative analysis of ECC, AES and RSA algorithms experimentally with parameters such as computation time and complexity of the algorithms. Based on the result, it was concluded that ECC has more complexity when compared to AES and RSA in mobile devices [43]. Giripunje et al., provided effective security solution using PKC and its implementation in two parts: first, design for API for ECC which generates shared key for secure communication and secondly, a web service is created which distributes this key to validate the mobile user [44].

Ameya Nayak discussed the growing android community, its malware attacks, security concerns, aid in serving as the continuous challenges of identifying current, future vulnerabilities as well as incorporating security strategies against them and this focus on mobile devices [45]. Srikanth Pullela discussed the performance issues of handoffs, routing, etc. Then, he further addressed that security is another key issue, which needs to be considered when the communication channel is set up. Most of them are based on the public and PKC [46].

V. Gayoaso Martinez and L. Hernandez Encinas, have discussed the ECC, one of the options for protecting sensitive information. The latest version of the JAVA platform includes a cryptographic provider—SunEC which implements EC operations and protocols [47]. Muhammad Waseem Khan explained that short message service (SMS) is one of the frequently used mobile services with universal availability in all GSM networks. However, SMS does not have its own built-in mechanism to secure the transmitted data because security is not considered as a priority application for mobile devices and provides room for the secure SMS message communication [48].

Ram Ratan Ahirwal and Manoj Ahke have explained two different methods to encrypt and decrypt the message. They pointed out that the second method supports the system with more security than the first method because the sender computes the exponentiation function between the coordinates of the encryption algorithm and the receiver computes the inverse of the exponentiation function between the coordinates of the key in the decryption algorithm [49].

Sathish Kumar et al., discussed the usage of mobile hand-held devices that are used in an efficient way to deliver real time data to the users in the battle field military applications such as data confidentiality, authentication, etc., which are not readily offered by mobile environment. It is necessary to design and implement an energy efficient authentication protocol that accomplishes a high level of security with minimum energy consumption and proposed an energy efficient authentication protocol for mobile devices [50].

Hamed Khiabani et al., explained the wireless networking, mobile and embedded devices, other pervasive computing technologies to provide security. Security and privacy are the main concerns in mobile computing [51]. Seema P. Nakhate and R.M. Goudar have implemented a secured password based mutual authentication protocol for client–server computing using ECC framework which provides secure communication with the help of user email-id and mobile phone authentication device for mobile handheld device Such devices are mobile phones, PDA's, Palmtops and Smart cards [52].

V. Vishnu and R. Shobha discussed the security in WSN. They have applied dynamic election of Cluster Head (CH) mechanism and two evolutionary approaches SET-IBS and SET-IBOOS which provide security in data transmission and reduce data losses due to nodes failure, less residual energy selected in CH [29]. Tanmoy Kumar Bishoi et al., proposed an algorithm to encrypt the data using symmetric key encryption technique and can be improved by using variable length key [53].

M. Sujithra et al., has explained that the cryptographic algorithms are implemented and tested in Local as well as cloud environment, verified that the mobile data in cloud increases efficiently and AES algorithm performs better when compared with other algorithms in mean processing time but the combination of MD5 + ECC + AES algorithms qualify better than speed-up ratio [54].

Said Bouchkaren and Saiida Lazaar discussed secure data transmission via Internet and also they have designed and implemented a cryptosystem due to a number of iterations of encryption and decryption of data in blocks, using cellular automata and compared them with AES algorithm and proved that the new algorithm resists against statistical attacks, faster than AES-256, achieved good confusion and diffusion tests [55].

4 Proposed methodology

SSO based AC generation procedure is proposed here to use with RSA and ECC cryptographic algorithms. SSO based optimized AC is performed in iterative basis. The AC generation optimization starts with small numbers and

then move on towards the large numbers. Optimization is one time process so it will not affect the runtime while generating ACs for a set of numbers. Applying SSO to generate ACs require the following initializations.

$$\text{Let } s = l(n) \tag{15}$$

where l(n) is the smallest length AC for a number n and

$$\log_2(n) + \log_2(v(n)) - 2.13 \le l(n) \le \log_2(n)(1 + 0(1))/\log_2(\log_2(n)) \tag{16}$$

where v(n) is Hamming weight.

Therefore,

$$l(2n) \le l(n) + 1. \tag{17}$$

The elements of the minimum length AC are substituted for particles in SSO as explained in Fig. 1.

The elements of CP are CP_i, i = 1, 2,… n. The elements' search spaces also limited to simplify the optimization process. The first element CP_1 value is restricted with the value 1 because all AC should start with 1. The second element CP_2 is restricted to 2 as the doubled value of 1. The first two elements are not involved in optimization process at all. The third element CP_3 can be either 3 (2 + 1) or 4 (2 + 2). The fourth element CP_4 can be 4, 5, 6 or 8. Then, completing all epochs, SSO Optimized particle (element) values are: CP_1 = {1}, CP_2 = {2}, CP_3 = {3, 4}, CP_4 = {4, 5, 6, 8}, CP_5 = {6, 7, 8, 9, 10, 11, 12}, CP_6 = {8,9,10,11,12,13,14,15,1,17,18,19,20,22,24}

Two possibilities of SSO ACs accomplishment are shown in Fig. 2. For example, AC for 78, using binary method is 1 → 2 → 4 → 8 → 9 → 18 → 19 → 38 → 39 → 78 with ln (78) = 9. While running SSO optimization, some possible ACs for the value 78 with l(n) = 8 are:

1 2 3 5 8 13 26 39 78| 1 2 3 5 8 13 26 52 78| 1 2 3 5 10 13 26 39 78| 1 2 3 5 10 13 26 52 78| 1 2 3 6 7 13 26 39 78| 1 2 3 6 7 13 26 52 78| 1 2 3 6 9 15 24 39 78| 1 2 3 6 9 15 30 39 78| 1 2 3 6 9 18 21 39 78| 1 2 3 6 9 18 36 39 78| 1 2 3 6 9 18 36 42 78|.

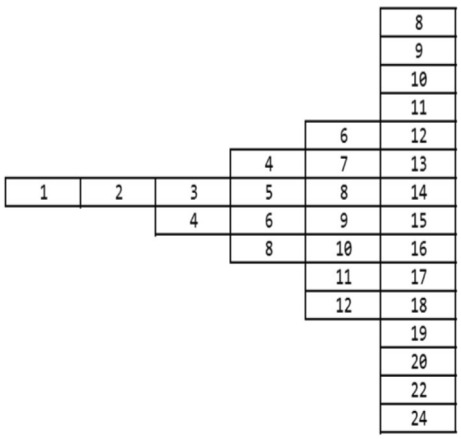


Fig. 1 The chain particles (CP)

After completing given epochs, SSO generates an optimized particle values in the element positions. The SSO optimized result particle values are given in Fig. 3. In SSO the optimization is started from the element CP_3 = {3,4} and it continues up to the element CP_{n-1}. While comparing SSOAC generation with binary AC generation method, SSO takes a little more processing time but the target of minimal length AC is achieved by SSOAC. The minimal length AC saves more time while calculating exponents in RSA and k[P] in ECC procedures so compromising a little more time. The AC for an integer n based on PSO is generated in this manner using the steps involved in PSO algorithm.

The flow diagram of the proposed methodology is shown in Fig. 4. First the emulator which is used for the choice of the device and then select the RSA/ECC for cryptographic algorithm. Then, finally select PSO/SSO with Addition Chain with RSA/ECC. This process can be repeatedly executed and stop when the final result (optimum solution) has been obtained.

5 Experimental set up

Two different types of mobile emulators are used in this work to measure the performance of various cryptography procedures. Widely used android and windows mobile devices are taken into experiment. To establish android OS and Windows OS mobile infrastructure [19], T-Engine Android emulator (A) and Windows Mobile (W) emulator 6.1.4 [56] respectively are used. A user interface (UI) is designed using Visual C++ to upload files and to measure emulator performance parameters for both Android and Windows Emulators. The UI and emulators executions are carried out in a 2.4 GHz Intel i5 processor-4 GB RAM computer with Windows 8.1 64-bit OS. Each emulator is launched individually to load and execute different encryption algorithms. The sample screenshots of UI, Android Emulator and Windows Mobile Emulator are shown in Figs. 5, 6, 7 respectively.

6 Results and discussions

Processing time, power consumption and security levels are the prime parameters to determine the quality of a cryptography algorithm. Encryption and decryption time refers the time taken to convert the plaintext into ciphertext and vice versa respectively. Both parameters get equal priority to measure the quality of a cryptography procedure. Lesser encryption and decryption time refers higher

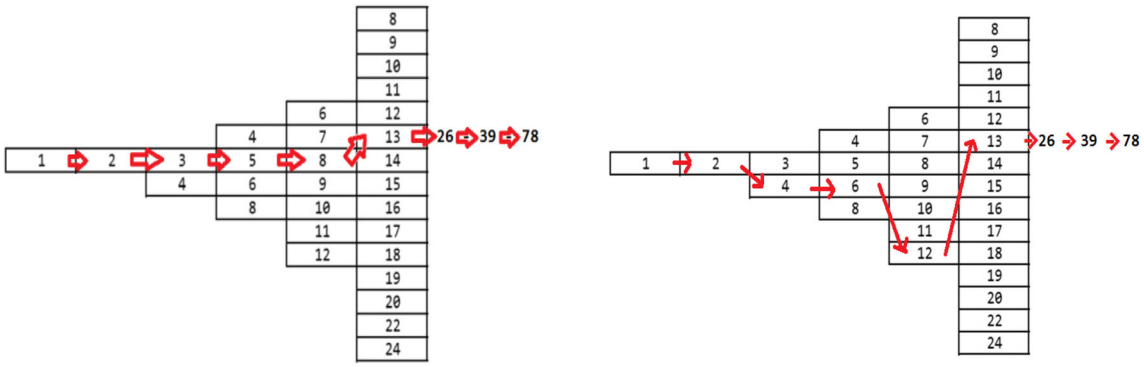


Fig. 2 Two different ACs for the integer 78 generated using SSO

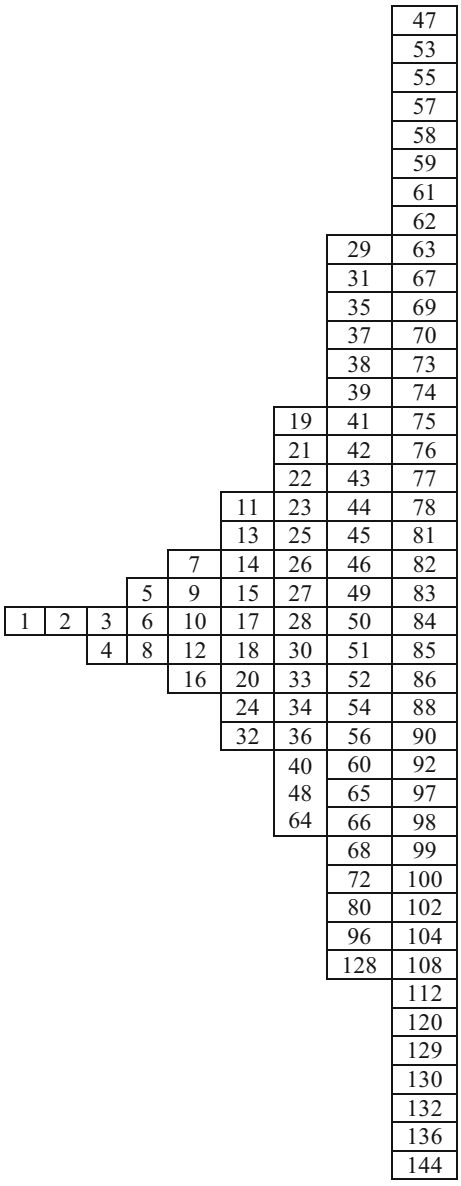


Fig. 3 SSO optimized result particle values

quality of the cryptography algorithm. Similarly, power consumed to encode a plaintext into ciphertext is called encryption power consumption. Mobile devices are battery powered devices and they provide greater mobility of the power is consumed cautiously. A good cryptography algorithm should be capable of processing with reasonable power consumption without compromising the security strength. Powers consumed by different cryptography methods for different file sizes are measured in android (A) and windows mobile (W) environment are compared.

Achieving higher security levels with lesser time and power consumption is the ultimate aim of an ideal cryptography procedure. Different file sizes 1 MB, 2 MB, 4 MB, 8 MB and 16 MB are used in this work to measure these parameters like encryption time, decryption time, power consumed for encryption, decryption and security levels are measured using All Block Cipher (ABC) Universal Hackman tool for RSA, ECC, RSA-PSOAC, ECC-PSOAC, RSA-SSOAC and ECC-SSOAC in both Android and Windows mobile. Measured results are recorded in tables. Table 1 shows encryption and decryption time using RSA in two different OS. Table 2 shows the time taken by their corresponding power consumption parameters.

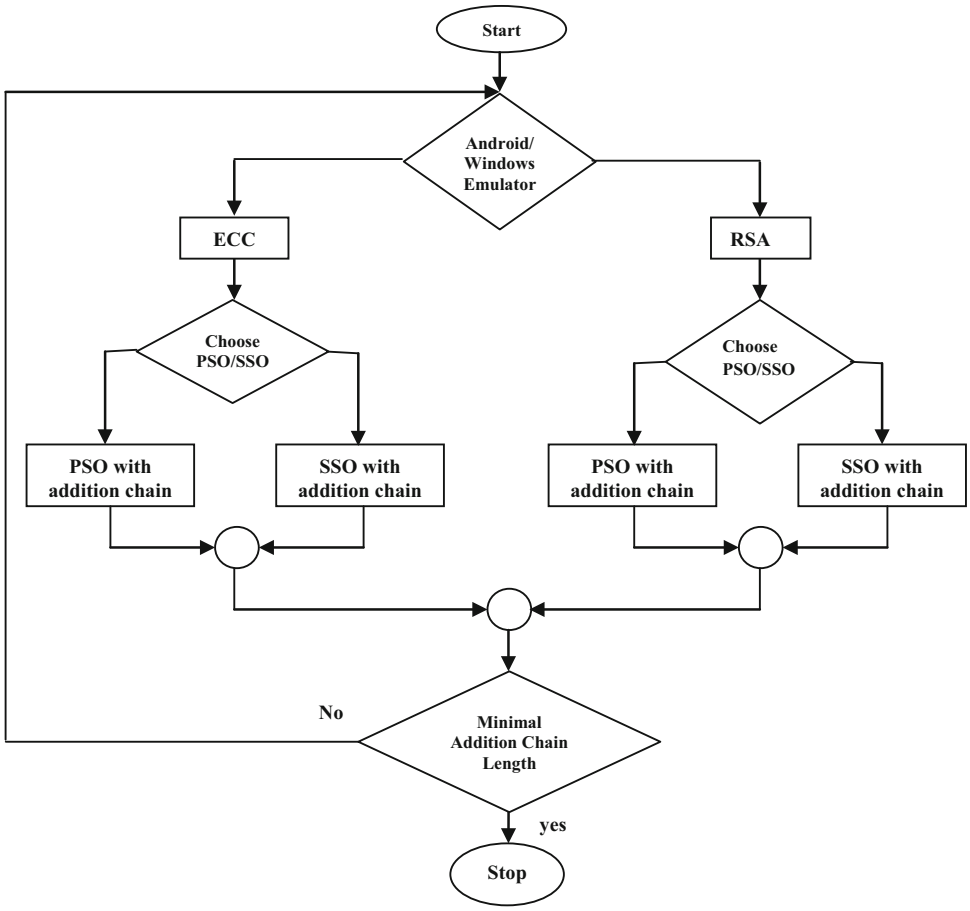
Table 3 shows encryption and decryption time using ECC in two different OS. Table 4 shows the time taken by their corresponding power consumption parameters.

Table 5 shows the time taken by encryption and decryption process after incorporating the AC based on PSO into RSA in two different OS. Table 6 shows time taken by their corresponding power consumption parameters.

Similarly, Table 7 shows the time taken by encryption and decryption process after incorporating the AC based on PSO into ECC in two different OS. Table 8 shows time taken by their corresponding power consumption parameters.

Table 9 shows the time taken by encryption and decryption process after incorporating the AC based on

Fig. 4 Flow diagram



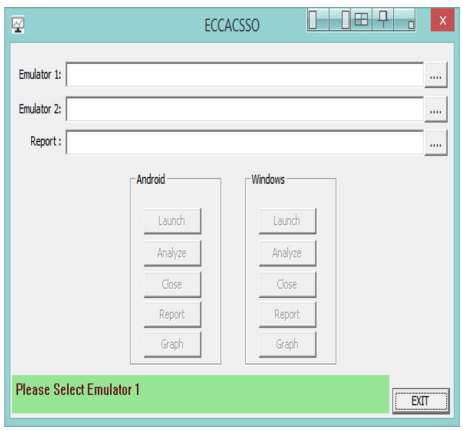


Fig. 5 User interface

SSO into RSA in two different OS. Table 10 shows time taken by their corresponding power consumption parameters.

Similarly, Table 11 shows the time taken by encryption and decryption process after incorporating the AC based on SSO into ECC in two different OS. Table 12 shows time taken by their corresponding power consumption parameters.

Fig. 6 Android emulator

Table 13 shows the security in RSA (Android) Vs RSA (Windows) and ECC (Android) vs ECC (Windows) and Table 14 shows the security in RSA-SOAC (Android) vs

Fig. 7 Windows mobile emulator

RSA-PSOAC (Windows) and ECC-PSOAC (Android) Vs ECC-PSOAC (Windows).

Table 15 shows the security in RSA-SSOAC (A) Vs RSA-SSOAC (W) & ECC-SSOAC (A) Vs ECC-SSOAC (W).

Figures 8 and 9 shows the security strength in Android as well as Windows environment.

As per the observations, ECC provides more security than RSA in both PSO and SSO combinations. RSA consumes lesser power than ECC in PSO and SSO combination. When security is concerned the ECC-SSOAC is recommended.

7 Results analysis

When transmitting any file securely it should be encrypted first. Based on two different OS's based emulators, the experimental results clearly reveal that the time taken for encryption and decryption, encryption and decryption

Table 1 Encryption time, decryption time (ms) RSA (A) vs RSA (W)

FS (MB)	RSA (A) ET	RSA (W) ET	RSA (A) DT	RSA (W) DT
1	1656	1628	571	541
2	3230	3180	1081	1073
4	6496	6490	2171	2162
8	13,691	13,657	4565	4554
16	27,434	27,422	9165	9165
Avg	10,501	10,475	3511	3499

FS file size, *A* Android, *W* Windows, *ET* encryption time (in ms), *DT* decryption time (in ms), *Avg* average time (in ms)

Table 2 Encryption power, decryption power (mW) RSA (A) Vs RSA (W)

FS (MB)	RSA (A) EP	RSA (W) EP	RSA (A) DP	RSA (W) DP
1	1658	1643	554	573
2	3218	3207	1081	1083
4	6484	6484	2177	2186
8	13,702	13,659	4577	4554
16	27,447	27,401	9165	9148
Avg	10,502	10,479	3511	3509

EP encryption power (in W), *DP* decryption power (in W)

Table 3 Encryption time (ms), decryption time ECC (A) vs ECC (W)

FS (MB)	ECC (A) ET	ECC (W) ET	ECC (A) DT	ECC [W] DT
1	2447	2292	856	791
2	4768	4541	1629	1512
4	9550	9220	3222	3083
8	20,185	19,413	6770	6485
16	40,420	38,961	13,493	12,997
Avg	15,474	14,885	5194	4974

Table 4 Encryption power, decryption power (mW) ECC (A) vs ECC (W)

FS (MB)	ECC (A) EP	ECC (W) EP	ECC (A) DP	ECC (W) DP
1	2452	2330	859	788
2	4748	4557	1590	1536
4	9562	9209	3194	3091
8	20,174	19,394	6750	6469
16	40,434	38,939	13,476	12,995
Avg	15,474	14,886	5174	4976

Table 5 Encryption time, decryption time (ms) RSA-PSOAC (A) vs RSA-PSOAC (W)

FS (MB)	RSA-PSOAC (A) ET	RSA-PSOAC (W) ET	RSA-PSOAC (A) DT	RSA-PSOAC (W) DT
1	1048	1035	363	352
2	2053	2008	692	673
4	4100	4094	1381	1374
8	8643	8624	2896	2881
16	17,326	17,310	5781	5781
Avg	6634	6614	2223	2212

Table 6 Encryption power, decryption power (mW) RSA-PSOAC (A) vs RSA-PSOAC (W)

FS (MB)	RSA-PSOAC (A) EP	RSA-PSOAC (W) EP	RSA-PSOAC (A) DP	RSA-PSOAC (W) DP
1	1040	1023	363	354
2	2040	2012	696	676
4	4098	4094	1382	1375
8	8647	8631	2887	2894
16	17,317	17,308	5785	5786
Avg	6628	6614	2223	2217

Table 7 Encryption time, decryption time (ms) ECC-PSOAC (A) vs ECC-PSOAC (W)

FS (MB)	ECC-PSOAC (A) ET	ECC-PSOAC (W) ET	ECC-PSOAC (A) DT	ECC-PSOAC (W) DT
1	1571	1533	543	525
2	3063	3020	1045	1017
4	6139	6145	2050	2055
8	12,970	12,918	4323	4318
16	26,001	25,970	8694	8681
Avg	9949	9917	3331	3319

power of RSA and ECC using Android OS takes more time than RSA using Windows OS. And the time taken for encryption, decryption, encryption power and decryption power of PSO with addition chain of RSA and ECC using Android OS takes more time than RSA RSA-PSOAC(W). Similarly the time taken for encryption, decryption, encryption power and decryption power of SSO with addition chain of RSA and ECC using Android OS takes more time than RSA RSA-SSOAC(W). Generally ECC taken more time than RSA and it is also proved here that the ECC(A) takes more time than RSA(A) and also the ECC-PSOAC(A) takes more time than RSA-PSOAC(A).

Table 8 Encryption power, decryption power (mW) ECC-PSOAC (A) vs ECC-PSOAC (W)

FS (MB)	ECC-PSOAC (A) EP	ECC–PSOAC (W) EP	ECC–PSOAC (A) DP	ECC-PSOAC (W) DP
1	1555	1533	532	529
2	3052	3015	1024	1013
4	6156	6150	2066	2064
8	12,949	12,925	4332	4316
16	26,001	25,952	8677	8659
Avg	9943	9915	3326	3316

Table 9 Encryption time, decryption time (ms) RSA-SSOAC (A) vs RSA-SSOAC (W)

FS (MB)	RSA-SSOAC (A) ET	RSA-SSOAC (W) ET	RSA-SSOAC (A) DT	RSA-SSOAC (W) DT
1	866	865	302	293
2	1694	1673	573	562
4	3415	3417	1141	1144
8	7211	7185	2418	2398
16	14,436	14,424	4827	4812
Avg	5524	5513	1852	1842

Table 10 Encryption power, decryption power (mW) RSA-SSOAC (A) vs RSA-SSOAC (W)

FS (MB)	RSA-SSOAC (A) EP	RSA-SSOAC (W) EP	RSA-SSOAC (A) DP	RSA-SSOAC (W) DP
1	865	850	288	293
2	1698	1692	577	570
4	3413	3412	1151	1150
8	7202	7175	2410	2394
16	14,436	14,422	4821	4812
Avg	5523	5510	1849	1844

Table 11 Encryption time, decryption time (mS) ECC-SSOAC (A) vs ECC-SSOAC (W)

FS (MB)	ECC-SSOAC (A) ET	ECC-SSOAC (W) ET	ECC-SSOAC (A) DT	ECC-SSOAC (W) DT
	1299	1302	422	422
2	2542	2520	800	792
4	5119	5125	1598	1596
8	10,819	10,767	3369	3361
16	21,666	21,627	6743	6734
Avg	8289	8268	2586	2581

And the ECC-SSOAC(A) takes more time than RSA-SSOAC(A). Hence it is proved that RSA-SSOAC is taking less time than any of the above method.

8 Conclusion and future extension

PSO and SSO based ACs are thought of incorporated into RSA and ECC and implemented successfully. Proposed SSO optimized AC based RSA consumes lower power than

Table 12 Encryption power, decryption power (mW) ECC-SSOAC (A) vs ECC-SSOAC (W)

FS (MB)	ECC-SSOAC (A) EP	ECC-SSOAC (W) EP	ECC-SSOAC (A) DP	ECC-SSOAC (W) DP
1	1293	1300	414	415
2	2548	2527	803	786
4	5122	5122	1598	1594
8	10,807	10,762	3375	3348
16	216,664	21,643	6745	6742
Avg	8287	8271	2587	2577

Table 13 Security (%) RSA (A) vs RSA (W) and ECC (A) vs ECC (W)

FS (MB)	RSA (A)	RSA (W)	ECC (A)	ECC (W)
1	92	93	93	94
2	90	90	91	92
4	88	88	91	91
8	87	87	90	89
16	86	86	88	88
Avg	88.6	88.8	90.6	90.8

Table 14 Security(%) RSA-PSOAC(A) Vs RSA-PSOAC(W) & ECC-PSOAC(A) Vs ECC-PSOAC(W)

FS (MB)	RSA-PSOAC (A)	RSA-PSOAC (W)	ECC-PSOAC (A)	ECC-PSOAC (W)
1	95	93	95	95
2	92	92	94	93
4	90	91	92	93
8	89	89	91	91
16	89	88	91	91
Avg	91	90.6	92.6	92.6

Table 15 Security (%) RSA-SSOAC (A) vs RSA-SSOAC (W) and ECC-SSOAC (A) vs ECC-SSOAC (W)

FS (MB)	RSA-SSOAC (A)	RSA-SSOAC (W)	ECC-SSOAC (A)	ECC-SSOAC (W)
1	93	93	95	97
2	92	91	94	93
4	91	90	92	92
8	90	89	92	92
16	89	88	91	91
Avg	91	90.2	92.8	93

any other procedures compared. When considering security, SSO optimized AC based ECC provides more security levels. So, it is recommended to use RSA-SSOAC when a mobile device has limited power source to operate. When security is concerned ECC-SSOAC provides more security level of 97%. Experimental results clearly revealed that the proposed SSOAC optimization with RSA and ECC cryptography systems can be used either to reduce operational power or to achieve improved security levels which are the prime motive of this paper. This work can be extended by using this concept in mobile cloud computing due to the serious limitations of memory space, battery power for energy as well as resource optimization techniques without compromising the security.

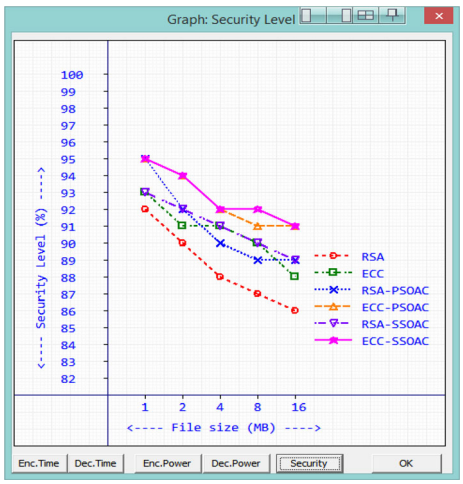


Fig. 8 Security strength (%) (Android)

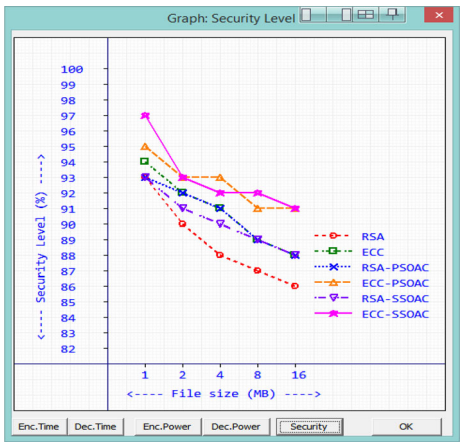


Fig. 9 Security strength (%) (Windows Mobile)

References

1. Meneses F, Fuertes W, Sancho J, Salvador S, Flores D, Aules H, Castro F, Torres J, Miranda A, Nuela D (2016) RSA encryption algorithm optimization to improve performance and security level of network messages. Int J Comput Sci Netw Secur
2. Bos JW, Halderman JA, Heninger N, Moore J, Naehrig M, Wustrow E (2014) Elliptic curve cryptography in practice. International conference on financial cryptography and data security. Springer, New York. https://eprint.iacr.org/2013/734.pdf
3. Renes J, Costello C, Batina L (2016) Complete addition formulas for prime order elliptic curves. In: Annual international conference on the theory and applications of cryptographic techniques. Springer, New York. https://eprint.iacr.org/2015/1060.pdf
4. Smart NP (2015) Elliptic curves. Cryptography made simple. Springer, New York
5. Katz NM, Mazur B (2016) Arithmetic moduli of elliptic curves. Annals of Mathematic Studies. Princeton University Press, Princeton
6. Paul1 T, Steve T (2017) Addition chains: a reSolve lesson. Austr Sr Math J. https://search.informit.com.au/documentSummary;dn=062208249066592;res=IELHSS
7. Li L, Li S (2017) Fast inversion in GF(2m) with polynomial basis using optimal addition chains. Circuits and systems (ISCAS). IEEE. https://ieeexplore.ieee.org/document/8050627/footnotes#footnotes
8. Du KL, Swamy MNS (2016) Particle Swarm Optimization. Search and optimization by metaheuristics. Springer, New York
9. Picek S, Coello CAC, Jakobovic D, Mentens N (2016) Evolutionary algorithms for finding short addition chains: going the distance. Evolutionary computation in combinatorial optimization. Springer, New York. https://link.springer.com/chapter/10.1007/978-3-319-30698-8_9
10. Picek S, Coello CAC, Jakobovic D, Mentens N (2017) Finding short and implementation-friendly addition chains with evolutionary algorithms. J Heuristics. https://dspace.mit.edu/handle/1721.1/115968
11. Mani K, Viswambari M (2017) A new method of generating optimal addition chain based on graph. Int J Math Sci Comput MECS
12. Couceiro M, Ghamisi P (2015) Particle Swarm Optimization. Fractional Order Darwinian Particle Swarm Optimization. Springer, New York
13. Yeh WC, Luo CY, Lai CM, Hsu CT, Chung YY, Lin JS (2016) Simplified swarm optimization with modular search for the general multi-level redundancy allocation problem in series-parallel systems. Evolutionary computation (CEC). IEEE. https://www.researchgate.net/publication/311254679
14. Stallings W (2006) Cryptography and network security principles and practices, 4th edn. Pearson Education Inc.,
15. Koblitz N (1994) A course in number theory and cryptography. Graduate texts in mathematics, 2nd edn. Springer, New York. https://www.springer.com/gp/book/9780387942933
16. Koblitz N (1987) Elliptic curve cryptosystems. Mathematics of Computation, vol 48, no 177. http://pages.cs.wisc.edu/~cs812-1/koblitz87.pdf
17. Miller V (1986) Use of elliptic curves in cryptography. Advances in Cryptology—CRYPTO'85, Lecture Notes in Computer Science. https://link.springer.com/chapter/10.1007/3-540-39799-X_31
18. Koziel B, Azarderakhsh R, Jao D, Mozaffari-Kermani M (2016) On fast calculation of addition chains for isogeny-based cryptography. Information Security and Cryptology
19. Acharya S, Shenoy a, Lewis M, Desai N (2016) Analysis and prediction of application usage in android phones. Advances in electrical, electronics, information, communication and bio-informatics. IEEE. https://ieeexplore.ieee.org/document/7538346
20. Pluhacek M, Janostik J, Senkerik R, Zelinka I, Davendra D (2016) PSO as complex network—capturing the inner dynamics—initial study. In: Proceedings of the Second International Afro-European Conference for Industrial Advancement. Springer, New York. https://www.springerprofessional.de/en/pso-as-complex-network-capturing-the-inner-dynamics-initial-stud/7383416
21. Meng XB, Gao XZ, Lu L, Liu Y, Zhang H (2016) A new bio-inspired optimisation algorithm: bird Swarm Algorithm. J Exp Theor Artif Intell. https://www.tandfonline.com
22. Liu Y, Li C, Wu X, Zeng Q, Liu R, Huang T (2016) Particle Swarm Optimizer with full information. Intelligent computing theories and application. Springer, New York
23. Yeh WC, Lin WT, Lai CM, Lee YC, Chung YY, Lin JS (2016) Application of simplified swarm optimization algorithm in deteriorate supply chain network problem. Evolutionary computation (CEC). IEEE. https://ieeexplore.ieee.org/document/7744127

24. Mavridis I., Pangalos G (1997) Security issues in mobile computing paradigm. https://link.springer.com/chapter/10.1007/978-0-387-35256-5_5
25. Olson E, Yu W (2000) Encryption for mobile computing
26. Chou W (2000) Elliptic curve cryptography and its applications to mobile devices. https://www.semanticscholar.org/paper
27. Elbaz L (2002) Using public key cryptography in mobile phones. White Paper, Discretix Technologies Ltd., Advanced security solutions for constrained environments. https://www.scribd.com/document/55521438/
28. Agrawal DP et al (2003) Secure mobile computing. In: Das SR, Das SK (eds) WDC. Springer, LNCS, New York. https://link.springer.com/chapter/10.1007/978-3-540-24604-6_26
29. Lufei H, Shi W (2006) An adaptive encryption protocol in mobile computing. Wireless/mobile network security. Springer, New York
30. Gupta AK (2008) Challenges of mobile computing. In: Proceedings of 2nd National Conference on Challenges & Opportunities in Information Technology RIMT—IET, Mandi Gobindgarth
31. Rao SKM, Reddy AV (2009) Data dissemination in mobile computing environment, vol 1, no 1. BIJIT, Bharati Vidyapeeth's Institute of Computer applications and Management (BVICAM), New Delhi. http://bvicam.ac.in/bjit/downloads/pdf/issue1/10.pdf
32. Doomun MR, Soyjaudah KMS (2009) Analytical comparison of cryptographic techniques for resource-constrained wireless security. Int J Netw Secur 9(1):82–94. http://ijns.jalaxy.com.tw/contents/ijns-v9-n1/ijns-2009-v9-n1-p82-94.pdf
33. Kar J, Majhi B (2009) An efficient password security of multi-party key exchange protocol based on ECDLP. Int J Comput Sci Secur 1(5):405–413
34. Kim M et al (2009) Design of cryptographic hardware architecture for mobile computing. J Inf Process Syst 5(4):187–196
35. Rocha BPS et al (2010) Adaptive security protocol selection for mobile computing. J Netw Comput Appl 33:569–587
36. Kumar SA (2010) Classification and review of security schemes in mobile computing. Wirel Sens Netw 24:419
37. Al-Bakri SH, Alam GM et al (2011) Securing peer-to-peer mobile communications using public key cryptography: new security strategy. Int J Phys Sci 6(4):930–938
38. Afreen R, Mehrotra SC (2011) A review on elliptic curve cryptography for embedded systems. Int J Comput Sci Inf Technol 3(3)
39. Rifa-Pous H, Herrera-Joancomarti J (2011) Computational and energy costs of cryptographic algorithms on handheld devices. Future Internet. https://doi.org/10.3390/fi3010031. https://www.mdpi.com/1999-5903/3/1/31 **(ISSN: 1999-5903)**
40. Bhatta J, Pandey LP (2011) Performance evaluation of RSA variants and elliptic curve cryptography on handheld devices. IJCSNS Int J Comput Sci Netw Secur 11(11):8
41. Kumar KS et al (2012) An experimental study on energy consumption of cryptographic algorithms for mobile hand-held devices. Int J Comput Appl 40(1):1–7
42. Nosrati M et al (2012) Mobile and operating systems. Comput Princ Dev World Appl Progr 2(7)
43. Mann RS et al (2012) A comparative evaluation of cryptographic algorithms. Int J Comput Technol Appl 3(5). https://pdfs.semanticscholar.org/9d7a/5ba194b3aafaf55e8db42e699b2a09832a4c.pdf
44. Giripunje L, Nimbhorkar S (2013) Comprehensive security system for mobile network using elliptic curve cryptography over GF (p). Int J Adv Res Comput Sci Softw Eng 3(5). https://www.semanticscholar.org/paper/
45. Nayak A (2013) Android mobile platform security and malware survey. IJRET 2(11). https://www.academia.edu/
46. Pullela S (2013) Security issues in mobile computing. Int J Res Eng Technol 2(11). https://pdfs.semanticscholar.org/2035/f3a467dcc3523c6c2454605c021aff9a353d.pdf
47. Martinez G, Encinas LH (2013) Implementing ECC with Java Standard Edition 7. Int J Comput Sci Artif Intell 3(4):134. http://www.academicpub.org/ijcsai/paperInfo.aspx?paperid=14496
48. Khan MW (2013) *S*MS security in mobile devices: a survey. Int J Adv Netw Appl 5(2):1873
49. Ahirwal RR, Ahke M (2013) Elliptic curve Diffie-Hellman key exchange algorithm for securing hypertext information on wide area network. Int J Comput Sci Inf Technol 4(2):363–368
50. Sathish K et al (2014) An asymmetric authentication protocol for mobile hand held devices using ECC over point multiplication method. Int J Adv Res Comput Sci Technol 2:393–399
51. Khiabani H et al (2014) A review on privacy, security and trust issues in mobile computing. Collaborative outcome of University of Malaysia and MIMOS Berhad, Information Security Cluster
52. Nakhate SP, Goudar RM (2014) Secure authentication protocol. Int J Comput Netw Commun Secur 2(4)
53. Vishnu V, Shobha R (2015) Dynamic cluster head (CH) node election and secure data transaction in CWSNs. Int J Eng Res 4(4). https://www.academia.edu/27582814/
54. Bishoi TK et al (2015) An algorithm on text based security in modern cryptography. J Comput Netw Wirel Mobile Commun 5(1)
55. Sujithra M et al (2015) Mobile data security: a cryptographic approach by outsourcing mobile data to cloud. Procedia Comput Sci 47:480–485
56. (2019) Emulator available at: https://www.microsoft.com/en-us/download/details.aspx?id=53424
57. Bouchkaren S, Lazaar S (2016) A new iterative secret key cryptosystem based on reversible and irreversible cellular automata. Int J Netw Secur 18(2):345–353

International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 /doi:10.14445/22315381/IJETT-V69I2P205
Volume 69 Issue 2, 32-38, February 2021

Generation of Addition Chain using Bacteria Foraging Optimization Algorithm

Dr.K.Mani[1], A. Mullai[2]

[1]*Associate Professor in Computer Science, Nehru Memorial College(Autonomous), Puthanampatti, Affiliated to Bharathidasan University, Tiruchirappalli, Tamil Nadu, India.*

[2]*Associate Professor in Computer Science, Seethalakshmi Ramaswami College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, Tamil Nadu, India.*

[1]nitishmanik@gmail.com, [2]mullai_a@yahoo.com

Abstract

In many number-theoretic cryptographic algorithms, encryption and decryption are of the form $x^n \bmod p$ where x, n and p are integers. When exponentiation operation is involved in many cryptosystems, it takes more time than any normal arithmetic operations. The computation time can be reduced by using repeated multiplications rather than using exponential operation. This can also be further reduced by using addition chain. Modular exponentiation with addition chain is used to determine the correct sequence of multiplications. There exist several algorithms in the literature to generate the optimal addition chain for the given integer. A novel bacteria foraging optimization algorithm based addition chain has been proposed and it is verified with the existing state of art of addition chain algorithms like genetic algorithm, evolutionary programming etc., in this paper.

Keywords *- Addition Chain, RSA, ECC, PSO, SSO, BFOA, Optimization.*

I. INTRODUCTION

An Addition Chain (AC) can be thought of as a sequence of integers in which the first number is always 1 and the last number is always n, where n is an integer for which ACs are to be generated. For finite fields, operations such as square roots or inversions, exponentiations can be performed efficiently by utilizing an optimal AC, the smallest such AC sequence to reach n. In particular, fast exponentiation and inversion are paramount to the performance of scalar point multiplication $k[P]$ where k is a scalar and P is a point in elliptic curve (EC) in elliptic curve cryptography (ECC) [1] [2], pairings in pairing-based cryptosystems, and computing isogenies in the quantum-resistant isogeny-based cryptosystems [23]. To get the next number, there are two steps normally used in AC. They are addition and doubling steps, i.e., to get the next number (intermediate number) in AC, any two previous numbers are added together in addition step, whereas in the doubling step, the current number is multiplied by two. To generate the AC for given n, two types of algorithms are normally used viz., deterministic and stochastic or bio-inspired.

In deterministic algorithms, since everything is deterministic and the optimal AC may not be obtained at all times. The binary method, factor method, window method, sliding window method, Fibonacci method, Lucas method, continuous fraction method, etc., are examples of the deterministic algorithm. Evolutionary algorithms or bio-inspired are inspired by the idea of either natural evolution or social behavior of insects or birds. The optimal ACs produced by evolutionary algorithms are not obtained by a single run. Many more runs are needed to obtain optimal AC, which will eventually take more times. Some examples of evolutionary algorithms are Genetic Algorithm(GA), Artificial Immune System(AIS), Ant Colony Optimization(ACO), Particle Swarm Optimization (PSO), Simplified Swarm Optimization(SSO), etc. Generating optimal AC for the given integer is an NP-hard problem because too many optimal ACs are generated. For example, different possible optimal ACs for the number 21 with length i.e., l(21) = 6 are:

1-2-3-4-7-14-21	1-2-3-6-9-15-21	1-2-4-5-10-20-21	1-2-3-6-9-12-21
1-2-3-5-7 -14-21	1-2-3-6-9-18-21	1-2-4-8-9-12-21	1-2-4-8-12-13-21
1-2-3-5-8-13-21	1-2-3-6-12-15-21	1-2-4-8-9-13-21	1-2-4-8-12-20-21
1-2-3-5 -8-16-21	1-2-3-6-12-18-21	1-2-4-8-9-17-21	1-2-4-8-16-17-21
1-2-3-5-10-11-21	1-2-4-6-7-14-21	1-2-4-8-10-11-21	1-2-3-6-7-14-21
1-2-3-5-10-20-21	1-2-4-8-16-20-21	1-2-4-8-10-20-21	

This is because 7 can be obtained by adding (7 = 3 + 4, 7 = 2 + 5, 7 = 1 + 6), 8 can be obtained by adding (8 = 4 + 4, 8 = 3 + 5) etc.

Bacteria foraging is one of the optimization and evolutionary algorithms. Kevin M. Passino proposed it in 2000, and it has been widely accepted as a new nature-inspired optimization algorithm [15]. It is inspired by the social foraging behavior of Escherichia Coli, i.e., a bacteria present in the human intestine and has drawn many researcher's attention. The underlying biology behind foraging is locomotion. During the foraging of the real bacteria, locomotion can be performed by a set of the tensile flagella and optimization process is achieved by foraging behaviour of bacteria in bacterium seeks to maximize the energy obtained per unit time spent during foraging. Suppose the flagella are rotated in the clockwise direction by the bacterium. In that case, the flagellum pulls on the cells, which results in independent movement of flagella, and the bacterium tumbles with lesser numbers of tumbling. Swimming at a very fast rate of the bacterium is performed with the flagella moving in the counter-clockwise direction.

The foraging strategy of E.coli is achieved by four processes viz., chemotaxis, swarming, reproduction and dispersal. Chemotaxis is a process which simulates the movement of E.coli cell through swimming and tumbling via flagella. Movement of E.coli bacterium can be performed in two ways viz., (i) swim for some time in the same direction or tumble (ii) alternate between the swim and tumble for the entire lifetime. In the swarming process, a group of E.coli cells arranged themselves in a traveling ring by moving up the nutrient gradient when placed amidst a semisolid matrix with a single nutrient chemo-effecter. The healthy bacteria asexually split into two bacteria, which are then placed in the same location while the least healthy bacteria eventually die in the reproduction process. In the elimination and dispersal process, gradual or sudden changes in the local environment, i.e., the significant local rise of temperature or due to unavoidable events, all the bacteria in a region are killed, or a group is dispersed into the new location.

In BFOA, generally, the bacteria can be moved for a long distance in a friendly environment. When sufficient food they had, their length also increased and will break in the middle to form a replica of themselves in the presence of a suitable environment. In swarm intelligence concept, this chemotactic progress may be eliminated and also a group of bacteria can move on to some areas or introduce some others related to the occurrences environmental changes like the event of elimination- dispersal done in the real bacterial population (where all the bacteria in a region are killed or a group will be dispersed into a new part of the environment).

II. RELATED WORK

In [3], Hugo Volger presented several results on $l(n)$. In particular, they determined $l(n)$ for all n satisfying $l(n)\leq 3$ and proved $\lfloor log n \rfloor + 2 \leq l(n)$ for all n satisfying $s(n) \geq 3$, where $s(n)$ is the extended sum of digits of n. In [4], Y.H. Tsai and Y.H. Chin found some mathematical properties of the shortest-length AC for certain integers whose binary patterns meet some special forms; and the correctness of these properties was proved. In [5], Bergeron et al. proposed generating the shortest AC based on the continued fraction. They gave a general upper bound for the complexity of continued fraction methods as a chosen strategy function. Thus, the total number of operations required for the generation of an AC for all integers up to n was shown to be $(n\ log^2 n\gamma n)$, where γn is the complexity of computing the set of choices corresponding to the strategy and proved an analogy of the Scholz-Brauer conjecture.

In [6], F. Bergeron et al. generated a method of fast addition chains with a small length of positive integer n, using continued fraction up to 1000 obtained with optimal length, (with 29 exceptions optimal length plus one). A new algorithm of optimal ACs described in [8] and also faster than the best-known methods. It is applicable for single values and slower than the best-known methods. This does not require any pre-computed values and is considered suitable for finding optimal ACs for point values.

Bounds on sums of ACs and properties of optimal ACs are discussed in [9].The study results that the final step in an optimal AC of an even number always have doubling, and also the sum of an optimal AC for an odd number $n$is asymptotically nearly $5n^2$. In [10], Noboru Kunihiro and Hirosuke Yamamoto developed two systematic methods viz., run-length encoding (RLE) and hybrid for generating short AC. They proved that the hybrid method was far better than RLE with a reduced 8% of the AC length.

In[11], Nareli Cruz-Cortéset al. explored the usage of a GA approach for the problem of finding optimal (shortest) ACs for optimal field exponentiation computations. The GA heuristic presented in this work was capable of finding almost all the optimal ACs for any given fixed exponent $e$with $e<4096$. They found that our GA strategy's percentage error was within 0.4% of the optimal for all cases considered. In other words, for any given fixed exponent e with $e<4096$, they found that strategy was able to find the requested shortest AC in at least 99.6% of the cases. In [12], N. Cruz- Cortés et al. proposed an artificial immune system(AIS) to generate an optimal AC. In that paper, they dealt with the optimal computation of finite field exponentiation, which is a well-studied problem with many important applications in error-correcting codes and cryptography.

In [13], Raveen R. Gounder et al. discussed a new strategy for doubling-free (SPA-resistant) short addition-subtraction chain(GRASC) for an arbitrary integer by using a precise golden ratio. In this, 12% to 28% reduction was obtained in the average chain length compared to other doubling-free AC methods. In [14], Alejandro Le´on-Javier et al. discussed the PSO algorithm to find short ACs with different exponents.

In[16], Mohamed M. Abd. Eldayamet al. proposed an algorithm for shorter AC based on the window method with small width using 2's complement. They proved that the proposed algorithm was more efficient than the last result with a 20% minimum. In[17], S Domínguez-Isidro and E Mezura-Montes et al. proposed an algorithm using evolutionary programming to find the minimal length AC and the results obtained were more promising than the other nature-inspired metaheuristic approaches but with a lower number of evaluations per run. The proposed EP algorithm comprised the solution encoding with suitable fitness function and initial population, a mutation operator, and the survivor selection mechanism, and EP does not use other operators such as crossover nor additional mechanisms like parent selection in GAs.

In [18], a note an addition chain was presented. Niel Michael Clift [19] proved the perfect matches in the Scholz–Brauer conjecture l(2n −1) = l(n) + n − 1 for new values. The minimal sequence of minimal multiplications required for performing modular exponentiation using Brauer Chains' concept by GA discussed in [20].

In [21], K. Mani proposed division based AC to generate the optimal ACs for the small exponents, exactly matched with ACs generated by the latest methods. But, for some large exponents, there was a very small increase in chain length (at most three).

In [24], a survey of the AC problem for optimizing the AC was made and effectively applied to implement a public-key cryptosystem. Mani K and Viswambari M [25] implemented a new method for the generation of the AC using graph G(V,E) where in the G's vertices refer to the numbers in the AC and edges refer to the move from one to another number in the AC. They have proposed two methods viz., Graph-Based All Possible AC (GBAPAC) generated all possible optimum ACs for the given integer n and Graph-Based Minimal AC (GBMAC), which generated the minimum number of optimum ACs by considering mutually exclusive edges starting from every number and also proved with the conjectures like Scholz-Brauer.

In [26], P. Anuradha Kameswari and B. Ravitheja derived a Lucas AC for any integer n to obtain Lucas sequence $V_n(a, 1)$ and also proved that the computation of $V_n(a, 1)$ using this Lucas AC is based on $V_{x+y}(a, 1)$ for $x, y, x - y$ in the Lucas AC. In [27], Stjepan Picek et al. derived that the GA approach with an novel encoding using crossover and mutation operators to minimize the length of the ACs with respect to a given exponent. Aaron Hutchinson and Koray Karabina implemented algorithms[28], for multidimensional differential ACs and applied these chains to ECC. This algorithm has the unique key features using n dimension. With key efficiency cum security features like uniformity, parallelized, and differential addition formulas were adopted by allowing speed using precomputation cost and storage requirements.

Dustin Moody and Amadou Tall [29], derived minimal chains with low Hamming weight using addition-subtraction chains with Lucas addition-subtraction in using $\ell-(n)$ the minimal length n, and proved that $|\ell-(2n) - \ell-(n)| \leq 1$ for all integers n of $Hammingweight \leq 4$ to have arrived a conclusion that minimal addition-subtraction chains for low Hamming weight integers, with the consideration of odd integers. In [30], Hazem M. Bahig and Yasser Kotb implemented a new parallel algorithm to obtain minimal AC for n. The experimental studies on multicore systems revealed that this algorithm's run time worked faster than the sequential one and obtained the maximum speed up of 2.5 times than the best known sequential algorithm.

In [31], A. Mullai and K. Mani proposed Particle Swarm Optimization (PSO) and Simplified Swarm Optimization (SSO) with ACs in RSA and ECC with two emulators, android and window. The processing time, power consumption was taken for encryption, decryption process, and security of the above was analyzed and also proved that the SSOAC optimization with RSA to reduce operational power and SSOAC optimization with ECC for more security. Narendra Mohan [32], discussed in Wireless Sensor Networks (WSNs), to enhance the network lifetime and minimize the energy consumption in sink nodes contains additional resources like long-range antenna, powerful batteries, large memory. This should be achieved using Enhanced Emperor Penguin Optimization (EEPO) algorithm.

III. THEORETICAL BACKGROUND

This section describes some mathematical preliminaries required for AC.

Definition 3.1 (Addition Chain)

An AC [7] for a positive integer n is a sequence, $1 = a_0 \leq a_1 \leq \cdots \leq a_r = n$ such that each member after a_0 is the sum of two earlier (not necessarily distinct) ones. The number $l(n)$ is called the length of the AC. It is noted that if the value of n is relatively small, the exact value of $l(n)$ is known.

Definition 3.2 (Optimal Addition Chain)

An AC is optimal if its length is the smallest among all possible ACs. For example, 1− 2 − 3 − 6 − 12 − 13 is one of the optimal chains for 13, and with l(13) =5.

The construction [20] of each element of an AC is called a step. For an AC, $1 = a_0 \leq a_1 \leq \cdots \leq a_r = n$, the following steps are involved. Doubling step: $a_i = 2a_{i-1}$, $i > 0$. Non-doubling step: $a_i = a_j + a_k$, $i > j > k \geq 0$. The steps of the form $a_i = 2a_j$, $j \leq i - 2$ are defined as non-doubling steps.

Big step: $\lambda(a_i) = \lambda(a_{i-1}) + 1$.

Small step: $\lambda(a_i) = \lambda(a_{i-1})$.

Thus, the length of the AC, $l(n)$ can be split into two components as $l(n) = \lambda(n) + S(n)$. where $S(n)$ is the number of small steps in an optimal AC for n.

IV. BFOA_AC - PROPOSED METHODOLOGY

In the proposed methodology, the concept of BFOA is used to generate the optimum length AC for an integer n, which utilizes the foraging behaviors of bacteria. i.e., chemotaxis, swarming, reproduction, and elimination dispersal [15], are the four principal mechanisms used in BFOA. In this optimization, a virtual bacterium called search agent is one trial solution that moves on the functional surface to find the optimal length AC. The cost or fitness function is computed with a minimum length approach based on the nutrient concentration of the bacterium's immediate environment, searching for numbers in AC. The swarming step is not considered for the generation of AC in this method. The following notations are used in generating the optimal AC in this paper.

j	Index for the chemotactic step
k	Index for the reproduction step
i	Index for the elimination-dispersal event
S	Total number of the bacterium in the population
d	The dimension of the search space. Here, d = 1
S_w	The swarming length
RP_n	Number of reproduction steps
ED_n	Number of elimination-dispersal events
P_{ed}	Elimination-dispersal probability
$C(i)$	The magnitude of the next number in the random direction specified by the tumble

To generate the AC for any integer n, the first number is always 1, and the second number is 2, i.e., AC starts with $a_0 = 1$ and $a_1 = 2$ and last number $a_r = n$. Let $(i, k, l) = \{(j, k, l | i = 1,2, \ldots, S)$ represents each number in the AC in the population S at the j^{th} chemotactic, k^{th} reproduction, and l^{th} elimination-dispersal steps. It is noted that initially, the length of AC is taken as very large for the given integer n. Too many ACs are generated for n, but all ACs generated are not necessarily optimum. Moreover, the generation of optimal AC is an NP-hard problem. The prime steps used in BFOA related to generating the AC are as follows.

A. Search Space

Here, the search space is taken as one dimension(i.e.,d=1), and also the integer numbers are involved in generating AC for any n. Since the difference between intermediate numbers in AC is finite, the search space is also finite.

B. Chemotaxis

The movement of an E.coli cell through swimming and tumbling via flagella is simulated by the chemotaxis process. When a bacterium meets a favourable environment (rich in nutrients and noxious free), it will continue swimming in the same direction. When it meets an unfavorable environment, it will tumble, i.e., change its direction**.** In BFOA[22], E.coli can swim for a period of time in the same direction, or it may tumble and alternate between these two modes of operation for the entire life time. It is the most important step in determining the optimal AC for n. For AC generation, swimming and tumbling represent addition and doubling step, respectively. The goal is to move to let the bacterium search for the next number in the AC with minimal step.

a) Minimum Intermediate Number in AC

It is noted that the number of intermediate numbers between 2 and n should be minimum and it is obtained by a minimum number of steps as far as possible so that $l(n)$ could be minimized by considering all the directions (previous numbers) from the current bacterium position (present current number) can be chosen for the next step. Initially bacterium i is positioned at number 1, Let $m=0$ i.e., $a_0 = 1$. From 1, then it should move to 2. Now, $m = m + 1$ i.e., $a_1 = 2$, $AC \leftarrow 1 - 2$; $l(AC) = 1$. From 2, it can move to either 3 or 4, Now, $m = m + 1$

$$2a_{m-1} = a_m + a_0,\ i > j > k \geq 0 \quad \ldots(1)$$

$$AC \leftarrow AC || a_m \quad \ldots(2)$$

Now,

$$new\_l(AC) = old\_l(AC) + l \ \ or \ \ l(AC) = m \quad \ldots(3)$$

All the intermediate numbers obtained in this step are added to the minimal set Φ_{min},i.e., $\Phi_{min} = \{a_m\}$. A random intermediate number $\leq a_m$ is chosen from this set, and it indicates the direction of movement (i.e., from which AC starts) of bacterium$i$.

$$\Delta(i) = rand\{\ x \in \Phi_{min}\} \quad \ldots(4)$$

Let, (j, k, l) represents i^{th} bacterium with 1-dimensional vector represented as, 1,2, …, Sat j^{th} chromatic, k^{th} reproductive and l^{th} elimination-dispersal step. Let C(i) be the step size, which is taken as a unity because, from the current number in the AC, only one next number in the AC is generated based on previous numbers. Thus, the movement of the bacterium may be represented in the chemotaxis process as

$$(j+1, K) = \theta^i(j,k) + C(i)\frac{\Delta(i)}{\sqrt{\Delta i^T \Delta(i)}} \quad \ldots(5)$$

Where Δ indicates a vector in the random direction whose elements are [1, x].
The movement of the bacterium is explained with tree diagram as shown in fig. 1.

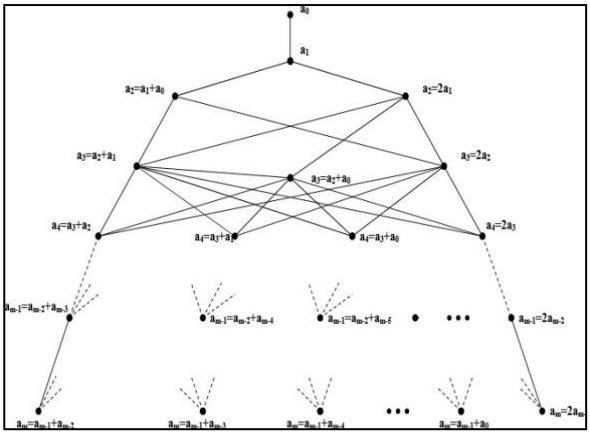


Fig.1: The Movement of Bacterium

C. Reproduction and Dispersal Step

Local search is provided by chemotaxis step, and the speed of convergence is achieved through the reproduction process. The bacteria which yields the maximal length of AC for n is called the least healthy bacteria, and it never produces the optimal length AC, which eventually dies. Each of the healthiest bacteria (yields minimum length AC) is asexually split into two bacteria, placed randomly. The dispersion process happens after a certain number of reproduction process. Depending on the probability, some bacteria were chosen to be killed or move to another position within the environment.

V. BFOA_AC – AN EXAMPLE

In order to understand the relevance of the work, let, n = 14, i = 1, m = 0, $a_m = a_0 = 1$ and initially bacteria b_1 is positioned at a_0. With the chemotaxis step, it moves to 2. Now, $m = m + 1$, i.e., $a_1 = 2$ and $l(a_1) = 1$. From a_1, b_1 moves to either 3 or 4 because $a_2 = a_1 + a_0 = 2 + 1 = 3$ or $a_2 = 2a_1 = 4$. Now, m = 2. Thus, Φ_{min}={3,4}. Let the intermediate number in AC randomly selected from Φ_{min}, i.e., $\Delta(1)$ = 3, Thus, the movement of b_1 is from 3, i.e., a_2=3 and the corresponding AC up to this stage is $1 - 2 - 3$ and $l(a_2) = 2$. From a_2, b_1 moves to either 4 or 5 or 6 because $a_3 = 2\,a_2 = 6$ or $a_3 = a_2 + a_0 = 3 + 1 = 4$ or $a_3 = a_2 + a_1 = 3 + 2 = 5$.Now, m = 3. Thus, Φ ={4,5,6}. Let 5 is selected randomly from the set Φ_{min}. Thus, $\Delta(1)$ = 5. The movement of b_1 is from 5, i.e., a_3 = 5. Correspondingly, AC up to this stage is $1 - 2 - 3 - 5$ and $l(a_3)$ = 3. From a_3, b_1 moves to either 6 or 7 or 8 or 10 because $a_4 = 2a_3 = 10$ or $a_4 = a_3 + a_0 = 5 + 1 = 6$ or $a_4 = a_3 + a_1 = 5 + 2 = 7$ or $a_4 = a_3 + a_2 = 5 + 3 = 8$. Now, m = 4. Thus, Φ_{min} = {6,7,8,10}. Let 7 is selected randomly from the set Φ_{min}. Thus, $\Delta(1)$ =7. The movement of b_1 is from 7, i.e., a_4 = 7. Correspondingly, AC up to this stage is $1 - 2 - 3 - 5 - 7$ and $l(a_4) = 4$. From a_4, b_1 moves to either 8 or 9 or 10 or 12 or 14 because $a_5 = a_4 + a_0 = 7 + 1 = 8$ or $a_5 = a_4 + a_1 = 7+2 = 9$ or $a_5 = a_4 + a_2 = 7+3 = 10$ or $a_5 = a_4 + a_3 = 7 + 5 = 12$ or $a_5 = 2(a_4) = 2(7) = 14$. Now, m = 5, Thus, Φ_{min} = {8,9,10,12,14}. Let 14 is selected randomly from the set Φ_{min}. Thus, $\Delta(1)$ = 14. The process is terminated because it reaches n = 17. Correspondingly, AC up to this stage is

$$1 - 2 - 3 - 5 - 7 - 14 \text{ and } l(a_5) = 5.$$

Suppose, other numbers from Φ_{min} are selected, even though it reaches 14 in the subsequent stages, l(14) is increased, and the corresponding bacteria will eventually die. Repeat the said process for other numbers, and the other ACs for 14 with l*(14) are given below.

1-2-3-4-7-14	1-2-4-5-7-14	1-2-4-6-8-14
1-2-3-5-7-14	1-2-4-5-9-14	1-2-4-6-10-14
1-2-3-6-7-14	1-2-4-5-10-14	1-2-4-6-12-14
1-2-3-6-8-14	1-2-4-6-7-14	1-2-4-8-10-14
1-2-3-6-12-14		1-2-4-8-12-14

VI. IMPLEMENTATION

The proposed methodology is implemented in VC++ and AC for the numbers up to 1024 are generated. It is shown in table 1. In table 1, l(r) indicates the sum of all optimal addition chains up to r. Table 1 exhibits the total l (up to 1024).

TABLE 1 TOTAL LENGTH OF OPTIMAL ADDITION CHAIN UPTO 1024

r	001-100	101-200	201-300	301-400	401-500	501-600	601-700	701-800	801-900	901-1000	1001-1024
l(r)	663	918	1011	1071	1121	1148	1183	1205	1230	1262	307
Total	4784					6028					307
Grand Total:11119											

Table 2 reveals AC generated for some hard exponents by BFOA where the hard exponent is the one for which AC is not easily found. Table 3 compares the optimal AC up to integers 1024 produced by the existing algorithms and the proposed BFOA.

TABLE 2 AC FOR HARD EXPONENTS BY BFOA

Exponents (E)	Optimal length (E)
2000	1 – 2 – 3 – 6 – 7 – 14 – 15 – 30 – 31 – 62 – 124 – 125 – 250 – 500 – 1000 – 2000.
2048	1 – 2 – 4 – 8 – 16 – 32 – 64 – 128 – 256 – 512 – 1024 - 2048.
4096	1 – 2 – 4 – 8 – 16 – 32 – 64 – 128 – 256 – 512 – 1024 – 2048 - 4096.
65131	1 – 2 – 3 – 5 – 7 – 11 – 19 – 29 – 47 – 71 – 127 – 191 – 379 – 607 – 1087 – 1903 – 3583 – 6271 – 11231 – 18287 – 34303 - 65131.
196591	1 - 2 - 3 - 5 - 7 - 11 - 19 - 29 - 47 - 71 - 127 - 191 - 379 - 607 - 1087 - 1903 - 3583 - 6271 - 11231 - 18287 - 34303 - 65131 - 110591 – 196591.
1176431	- 7 - 11 - 19 - 29 - 47 - 71 - 127 - 191 - 379 - 607 - 1087 - 1903 - 3583 - 6271 - 11231 - 18287 - 34303 - 65131 - 110591 - 196591 - 357887 - 685951 – 1176431.
2211837	1 - 2 - 3 - 6 - 9 - 15 - 30 - 60 - 120 - 126 - 252 - 504 - 1008 - 2016 - 4032 - 8062 - 16128 - 16143 - 32286 - 64572 - 129144 - 258288 - 516576 - 1033152 - 2066304 - 2195448 - 2211591 - 2211717 – 2211837.
4169527	1 - 2 - 3 - 5 - 7 - 11 - 19 - 29 - 47 - 71 - 127 - 191 - 379 - 607 - 1087 - 1903 - 3583 - 6271 - 11231 - 18287 - 34303 - 65131 - 110591 - 196591 - 357887 - 685951 - 1176431 - 2211837 – 4169527.
14143037	1 - 2 - 3 - 5 - 7 - 11 - 19 - 29 - 47 - 71 - 127 - 191 - 379 - 607 - 1087 - 1903 - 3583 - 6271 - 11231 - 18287 - 34303 - 65131 - 110591 - 196591 - 357887 - 685951 - 1176431 - 2211837 - 4169527 - 7624319 – 14143037.

From table 3, it is observed that the total length of optimal AC produced by BFOA with integers up to 1024 is 11119. They are almost the same as the optimal addition chains and their length produced by EP.

TABLE 3 COMPARISON OF AC UPTO INTEGERS 1024(PRODUCED BY EXISTING ALGORITHMS AND THE PROPOSED BFOA)

R	Opt.	AIS	GA	EP	BFOA
[1,512]	4924	4924(+)	4924	4924	4924
[1,1000]	10808	10813(+)	10813	10808	10812
[1.1024]	11115	11120(+)	-	11115	11119

VII. CONCLUSION

BFOA based AC has been thought of and it is implemented successfully. In this paper, ACs produced by some integers are proved both theoretically and experimentally. From the experimental results, up to integers 1024, the proposed BFOA algorithm produces the same optimal length AC which is almost equal to other existing evolutionary algorithms like AIS, GA, and EP. Further, the optimal length of AC for some hard exponents are the same as other existing evolutionary algorithms. This paper also provides an idea about the generation of AC based on BFOA. In future, this concept may be incorporated into public-key algorithms like RSA and ECC to reduce the encryption and decryption time because the said algorithms are used in mobile devices.

REFERENCES

[1] N Koblitz, Elliptic Curve Cryptosystems, Mathematics of Computation, 48(1982) 203-209.

[2] I Blake, G Seroussi and NP Smart, Elliptic Curves in Cryptography, Ser. London Math. Soc. Lecture Note Series, Cambridge Univ. Press,1999.

[3] Hugo Volger, Some Results on Addition/Subtraction Chains, Information Processing Letter, Elsevier, 1985.

[4] Y H TsaiandY H Chin, "A Study of Some Addition Chain Problems", International Journal of Computer Mathematics, 22(02) (1987) 117-134.

[5] R Begeron, J Berstel, S Brlek, and C Duboc, Addition Chains Using Continued Fractions, Journal of Algorithms, Elsevier, 10(1989) 403-412.

[6] Bergeron, JBerstel and S Brlek, Efficient Computation Of Addition Chains", Journalde Théorie des Nombresde Bordeaux, 6(1)(1994) 21-38.

[7] Donald E Knuth, The Art of Computer Programming, Seminumerical Algorithms, 2(3), Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA,1997.

[8] Gordon DM, A Survey of Fast Exponentiation Methods, Journal of Algorithms, 27(1998).

[9] H Zantema, Minimizing Sums of Addition Chains, Journal of Algorithms, Elsevier, 12(2) (1999) 281-307.

[10] Noboru Kunihiro and Hirosuke Yamamoto, New Methods for Generation of Short Addition Chains, IEICE Transactions Fundamental, 83(1)(2000).

[11] Nareli Cruz-Cortés, Francisco Rodriguez-Henriquez, RaúlJuárez-Morales and Carlos A Coello- Coello, Finding Optimal Addition Chains Using a Genetic Algorithm Approach, Springer- Verlag, (2005) 208-215.

[12] N Cruz-Cortes, F Rodriguez-Henriquez, and C A Coello-Coello, An Artificial Immune System Heuristic for Generating Short Addition Chains, IEEE Transactions on Evolutionary Computation, 6(2005) 252–280.

[13] Raveen R Goundar, Ken-ichiShiota, M Toyonaga,New Strategy for Doubling - Free Short Addition-Subtraction Chain, Mathematics,2008.

[14] AlejandroLe´on-Javier,NareliCruz-Cort´es,MarcoAMoreno-Armend´ariz,andSandraOrantes- Jim´enez, Finding Minimal Addition Chains with a Particle Swarm Optimization Algorithm, Advances in Artificial Intelligence, Springer, (2009) 680-691.

[15] Swagatam Das, ArijitBiswas, Sambarta Dasgupta, and Ajith Abraham, "Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications", Foundations of Computational Intelligence, Springerlink.com, Springer-Verlag Berlin Heidelberg, 3SCI 203(2009) 23–55.

[16] Mohamed M Abd-Eldayem, EhabT Alnfrawy, and AlyA Fahmya, Addition-Subtraction Chain for 160-bit Integers by using 2's Complex N Cruz-Cortés, F Rodríguez-Henríquez,and C A Coello-Coello, Addition Chain Length Minimization With Evolutionary Programming, Proceedings of Genetic and Evolutionary Computation Conference (GECCO) ACM digital Library, (2011).

[17] S Domínguez-Isidro and E Mezura-Montes, An Evolutionary Programming Algorithm to Find MinimalAdditionChains,ICongresoInternacionaldeIngenieríaElectrónica,Instrumentación y Computación, de Juniodel, Minatitlán Veracruz, México,2011.

[18] Maurice Mignotte, A Note on Addition Chains, International Journal of Algebra, 5(6)(2011).

[19] Neill Michael Clift, Calculating Optimal Addition Chains", Journal of Computing, Springer, 91 (2011) 265–284.

[20] Arturo Rodriguez-Cristerna and Jose Torres-Jimenez, A Genetic Algorithm for the Problem of Minimal Brauer Chains for Large Exponents, Soft Computing Applications in Optimization, Control, and Recognition, Springer, (2013) 27-5.

[21] K. Mani, Generation of Addition Chain using Deterministic Division Based Method, International Journal of Computer Science & Engineering Technology, 4(05) (2013) 553- 560.

[22] Om PrakashVerma, Rashmi Jain, and Vindhya Chhabra, Solution of Travelling Salesman Problem Using Bacteria Foraging Optimization Algorithm, International Journal of Swarm Intelligence, Inderscience publisher, 1(2) (2014).

[23] BrianKoziel, Reza Azarderakhsh, David Jaoand Mehran Mozaari-Kermani, On Fast Calculation of Addition Chains for Isogeny - Based Cryptography, Inscrypt 2016, IACR Cryptology, 2016.

[24] Adamu Muhammad Noma, Abdullah Muhammed, Mohamad Afendee Mohamed, and Zuriati Ahmad Zulkarnain. A Review on Heuristics for Addition Chain Problem: Towards Efficient Public-Key Cryptosystem, Journal of Computer Science, 13(2017) 275-289.

[25] K Mani, M Viswambari, A New Method of Generating Optimal Addition Chain Based on Graph, International Journal of Mathematical Sciences and Computing, 2(2017) 37-54.

[26] P Anuradha Kameswari and B Ravitheja, Addition Chain For Lucas Sequences With Fast Computation Method, International Journal of Applied Engineering Research, 13(11) (2018) 9413–9419.

[27] Stjepan Picek, Carlos A CoelloCoello, Domagoj Jakobovic and Nele Mentens, Finding Short And Implementation - Friendly Addition Chains With Evolutionary Algorithms, Journal of Heuristics, 24 (2018) 457-481.

[28] Aaron Hutchinson and Koray Karabina, Constructing Multidimensional Differential Addition Chains and Their Applications, Springer, Journal of Cryptographic Engineering, 9(2019) 1- 19.

[29] Dustin Moody and Amadou Tall, On Addition-Subtraction Chains of Numbers With Low Hamming Weight", Number Theory Mathematics, 25(2019) 155-168.

[30] Hazem M. Bahig· and Yasser Kotb, An Efficient Multicore Algorithm for Minimal Length Addition Chains, Computers, MDBI, 8(2019).

[31] A Mullai and K Mani, Enhancing The Security In RSA and Elliptic Curve Cryptography Based on Addition Chain Using Simplified Swarm Optimization and Particle Swarm Optimization For Mobile Devices, International Journal of Information Technology, Springer, (2020).

[32] Narendra Mohan Lifetime Enhancement of Sensor Nodes Based On Optimized Sink Node Placement Approach, International Journal of Engineering Trends and Technology 68.10(2020):10-23.

APPENDIX - A.3

SAMPLE CODING

```
// ACC.h: interface for the ACC class.

//Advanced C 8.1 [15052015]

//////////////////////////////////////////////////////////////////////


#if
!defined(AFX_ACC_H__FCCA8395_1892_4D17_AC1A_31C3FB2135A2__INCLU
DED_)

#define
AFX_ACC_H__FCCA8395_1892_4D17_AC1A_31C3FB2135A2__INCLUDED_


#if _MSC_VER > 1000

#pragma once

#endif // _MSC_VER > 1000


//HASH

#define DEF 0x00

#define SHA32 0x01

#define SHA64 0x02

#define SHA128 0x03

#define KHA 0x04


class AFX_EXT_CLASS ACC

{

public:

        void es(char s[1000],char t[2000],char x[10],int l);

        void GetProcessMessage(char me[200]);
```

```
char gps[100];
void BrowseAllFiles(char _fname[1000]);
void BrowseAllFolders(char _fold[1000]);
void BulkCopy(char _src[1000],char _des[1000]);
void FindBrowser(char _bname[1000]);
void FindAssociatedApplication(char _fn[1000],char _app[1000]);
bool fbc(char _ofn[1000],char _nfn[1000]);
void WaitDelete(char _fn[1000]);
long TimeDifference(SYSTEMTIME _t1,SYSTEMTIME _t2);
void ClosePrinterDC();
void StopPrinter();
bool StartPrinter();
bool SetPrinterDC(void);
CDC PrinterDC;
CDC *gwdc;
void ShowComboBox(CComboBox &_ccb);
void HideComboBox(CComboBox &_ccb);
void HideEditBox(CEdit &_ce);
void ShowEditBox(CEdit &_ce);
void DisableEditBox(CEdit &_ce);
void EnableEditBox(CEdit &_ce);
bool IsFilePresent(char _fn[1000]);
void SetFileToPath(char _fn[1000],char _path[1000],char _res[1000]);
bool CreateDumpFile(char _fn[1000],char _msg[1000]);
void RunAsApplication();
void RunAsService();
bool IsRunning(char _classname[50],char _windowname[150]);
```

```
CWnd* GetApplication(char _classname[50],char _windowname[150]);
void RGBtoColors(COLORREF _rgb,unsigned char &_r,unsigned char &_g,unsigned char &_b);
UINT CreateGraph();
bool bc(char _ofn[1000],char _nfn[1000]);
void Play(char _fn[1000],unsigned char _mode=0);
void PlayTime();
UINT CreateBluetooth();
UINT CreateDynamicHandle();
UINT CreateESP();
char RegisterInf(char fn[1000],char uid[50]);
void SetCalendar(CStatic &_cs);
void InitClock();
bool l1,l2,release;
void StopClock();
void SetClock(CStatic &_cs,char _m);
void SetClock(CStatic &_cs);
void EnableButton(CButton &_cb);
void DisableButton(CButton &_cb);
void ShowButton(CButton &_cb);
void HideButton(CButton &_cb);
void ShowStatic(CStatic &_cs);
void HideStatic(CStatic &_cs);
void GetTimeString(char _s[15],bool _h12);
void GetTimeString(char _s[15]);
void GetDateString(char _s[15]);
bool GetFileSize(char fn[1000], unsigned long long &fsize);
```

```
	bool CHash(unsigned char pro, char fname[1000], char hash[100]);
	SYSTEMTIME st;
	char s[500];
	ACC(HWND _hw);
	void Show();
	void Hide();
	void SetHandle(HWND _hw);
	ACC();
	virtual ~ACC();

private:
	DOCINFO gpdi;
	char gem[200];
	bool acr;
	HWND hw;
public:
	void SetGWDC(CDC* wdc);
	HICON LoadAppIconRes(UINT ICONRES);
	bool ExePro(char file[1000]);
	bool ExeProEx(char file[1000], char arg1[1000], char arg2[1000]);
};

#endif //
!defined(AFX_ACC_H__FCCA8395_1892_4D17_AC1A_31C3FB2135A2__INCLUDED_)

// BCCC.h: interface for the BCCC class.
// REV: AAZGAZBD
```

```
// REV: ZFZFAZBI
//////////////////////////////////////////////////////////////////


// This header file has methods for the following technologies
// Cryptography
// Data Security
// Data Compression
// Network Security
// Cloud Security
// Cloud offloading
// Mobile Cloud Offloading
// Data Integrity
// Grid Computing
// Parallel Processing
// Processor Scheduling
// Data Mining
// Big Data Analysis
// Optimizations


#if
!defined(AFX_BCCC_H__8B355BA0_920B_4999_83F1_7C8AB544DA29__INCL
UDED_)
#define
AFX_BCCC_H__8B355BA0_920B_4999_83F1_7C8AB544DA29__INCLUDED_


#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
```

```
//Cryptographic Procedures
#define RAC 0x00      //Random Access Cryptography        : Hardaware Version
#define ECC 0x01      //Elliptic Curve Cryptography
#define ELG 0x02      //ElGamal
#define NAF 0x03      //Non Adjacent Form
#define WNAF 0x04     //Windowed NAF
#define DSA 0x05      //Digital Signature Algorithm
#define SHA 0x06      //Secure Hash Algorithm
#define QCRY 0x07     //Quantum Cryptography : Hardware Version
#define RC4 0x08      //Rivest Cipher 4
#define RC5 0x09      //Rivest Cipher 5
#define RSA 0x0A      //Rivest Shamir Adleman
#define BCC 0x0B      //Block Cipher Cryptography : PCH version
#define AES_DFA 0x0C      //Advanced Encryption Standard Differential Fault
Analysis
#define DES_DFA 0x0D      //Data Endryption Standard Differential Fault Analysis
#define SRAC 0x0E         //Random Access Cryptography : Software Version
#define GSRAC 0x0F        //Genetic Random Access Cryptography : Software
Version
#define ECC_DH 0x10             //ECC with Diffie Hellman
#define GELG 0x11         //Genetic ElGamal
#define GNAF 0x12         //Genetic NAF
#define GWNAF 0x13              //Genetic WNAF
#define GDSA 0x14         //Genetic DSA
#define GSHA 0x15         //Genetic SHA
#define GECDHM 0x16             //Genetic ECC_DH
#define KNAPS 0x17        //Knapsack inherited Cryptography
```

```
#define MECC_DH 0x18      //Montgomery ECC_DH
#define MMECC_DH 0x19     //Modified Montgomery ECC_DH
#define AES 0x1A          //AES standalone
#define EDSSUMRT 0x1B     // Loaded from PCH
#define ECC_AC    0x1C    // ECC Addition Chain
#define ECC_ANNAC 0x1D    // ECC ANN Addition Chain
#define EVOTSCH 0x1E      //Voting Scheme
#define ECCRCV 0x1F       //CCRCV
#define ETHCM 0x20//Legacy
#define ECC_EAC 0x21      // ECC Enhanced Addition chain
#define DES 0x22      //      Data Encryption Standard
#define PPTDES 0x23       //      Primitive pythagorean Triples DES
#define PPTDDES 0x24      //      PPTDDES PPT Different Rotating Hamming Distance
#define BWTELG 0x25       //      Burrows Wheeler Transform ElGamal
#define EBWTELG 0x26      //      Enhanced Burrows Wheeler Transform ElGamal
#define MRRSA 0x27        // Magic Rectangle RSA
#define MRELG 0x28//      Magic Rectangle El-Gamal
#define BWTRSA 0x29       // Burrows Wheeler Transform RSA
#define EBWTRSA 0x2A      // Enhanced Burrows Wheeler Transform RSA
#define ECC_1AC 0x2B      // ECC 1's Complement Addition Chain
#define HIL 0x2C          // Hill Cipher
#define MHIL 0x2D         // Modified Hill Cipher
#define UDES 0x2E         // U-Matrix DES
#define URDES 0x2F        // U-Matrix Random Key DES
#define BFISH 0x30        // Blow Fish
#define AROMON 0x31             // User defined Legacy
```

#define SKM 0x32 // User defined Legacy

#define ECSRZ 0x33 // ElGamal Encryption using Elliptic Curve Spiral Clockwise Rotation and Zigzak Encryption Model

#define PPTRSA 0x34 // Primitive pythagorean Triples RSA

#define CLC 0x35 //Configurable Lattice Cryptography

#define CPEL 0x36 //Cantor Pair before ElGamal

#define RPEL 0x37 //Rosenberg Pair before ElGamal

#define EPEL 0x38 //Elegant Pair before ElGamal

#define ELCP 0x39 //ElGamal Cantor Pair

#define ELRP 0x3A //ElGamal Rosenberg Pair

#define ELEP 0x3B //ElGamal Elegant Pair

#define MRSA_2K 0x3C //Modified RSA-based algorithm A double secure approach 2 Key

#define MRSA_3K 0x3D //Modified RSA-based algorithm A double secure approach 3 Key

//Compression

#define NONE 0x00 //None

#define ZIP 0x01 //Zip

#define ZIP7 0x02 //7-Zip

#define RAR 0x03 //Real Archieve

#define HUFLAG 0x04 //Huffman Lagrange's

#define HUFFIB 0x05 //Huffman Fibbonacci

#define HUFLUC 0x06 //Huffman Lucas

#define HUFCOL 0x07 //Huffman Collective

#define RLE 0x08 //Run Length Encode

#define OPTRLE 0x09 //Orthogonal Polynomial Transform RLE

```
//Digital Signature Algorithms
#define DS_DEFA 0x00        //Default Hash Signature
#define DS_RSA 0x01             //Rivest Shamir Adleman
#define DS_ELG 0x02             //ElGamal
#define DS_DSA 0x03             //Digital Signature Algorithm
#define DS_ECDSA 0x04     //Elliptic Curve Digital Signature Algorithm
#define DS_GOST 0x05        //GOST R 34.10 - 2012
#define DS_SSA 0x06             //Schnorr Signature Algorithm
#define DS_RDS 0x07             //Rapid Digital Signature
#define DS_GMR 0x08             //Goldwasser Micali Rivest
#define DS_RCS 0x09             //Robin Crypto System
#define DS_EDDSA 0x0A     //Edwards-curve Digital Signature Algorithm
#define DS_ESIGN 0x0B      //Electronic Signature Algorithm
#define DS_MECDSA 0x0C //Modified Elliptic Curve Digital Signature Algorithm

//Variables
#define DEF 0x00      //Default
#define FSI 0x01      //File Size
#define KSI 0x02      //Key Size

//Optimizations
#define ACO 0x01     //Ant Colony Optimization
#define PSO 0x02     //Particle Swarm Optimization
#define SSO 0x03     //Simplified Swarm Optimization
#define TTSO 0x04    //Track Trade Spend Optimization
#define GNO 0x05     //Genetic Numeric Optimization
#define NACO 0x06   //Nodal Ant Colony Optimization
```

```
#define BFO 0x07      //Bacterial Forage Optimization
#define ADC 0x08      //Addition Chain

//Processor Architecture
#define HASWELL 0x01
#define S_BRIDGE 0x02       //Sandy Bridge
#define I_BRIDGE 0x03       //Ivy Bridge
#define T_BRIDGE 0x04       //Intel T-Bridge
#define X_BRIDGE 0x05       //Intel X-Bridge
#define Q_BRIDGE 0x06       //QualComm Multicore

//Processor Scheduling
#define GANG_FILL 0x01    //Gang-Fill
#define BACK_FILL 0x02    //Back-Fill
#define EEFF_FILL 0x03      //Energy Efficient Fill
#define MAXP_FILL 0x04    //Maximum Performance Fill

//File Types
#define UNKN 0x00
#define TEXTF 0x01
#define BINARY 0x02

// Data Mining
#define DDM 0x00 //DDM
#define APRI 0x01 //Apriori
#define FPTR 0x02 //FP-Tree
#define FPGR 0x03 //FP-Growth
```

APPENDIX - A.4
SAMPLE REPORTS

This Report is Generated by REAC App on 14-06-2021 at 12:32:52

Parameter: Encryption Time (mS) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC
1	1660	2447	1227	1820
2	3237	4790	2377	3555
4	6494	9553	4788	7173
8	13689	20179	10077	15130
16	27426	40415	20199	30326

Parameter: Decryption Time (mS) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC
1	1616	2330	1205	1789
2	3193	4519	2352	3519
4	6490	9209	4776	7171
8	13645	19375	10066	15082
16	27399	38936	20193	30294

Parameter: Encryption Power (mW) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC
1	554	817	421	613
2	1102	1621	806	1197
4	2171	3208	1612	2410
8	4569	6731	3364	5042
16	9156	13510	6735	10134

Parameter: Decryption Power (mW) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC
1	562	815	411	602
2	1081	1506	802	1178
4	2175	3072	1608	2417
8	4548	6463	3357	5035
16	9148	12995	6743	10109

Parameter: Security (%) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC
1	89	93	92	94
2	88	89	89	92
4	87	88	88	90
8	85	88	87	90
16	85	86	87	89

Parameter: Encryption Time (mS) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC
1	1654	2441	1204	1728
2	3233	4748	2378	3362
4	6490	9559	4775	6753
8	13670	20174044	10082	14252
16	27432	3	20214	28603

Parameter: Decryption Time (mS) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC
1	1616	2322	1205	1708
2	3184	4530	2366	3336
4	6496	9212	4778	6755
8	13657	19393895	10050	14240
16	27401	8	20210	28571

Parameter: Encryption Power (mW) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC
1	571	840	421	592
2	1100	1582	796	1146
4	2179	3228	1604	2263
8	4577	6750	3361	4765
16	9165	13496	6738	9559

Parameter: Decryption Power (mW) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC
1	541	785	415	584
2	1086	1536	805	1140
4	2192	3088	1604	2275
8	4579	6496	3368	4761

16	9133	12997	6742	9529

Parameter: Security (%) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC
1	91	92	92	94
2	88	89	89	92
4	86	89	88	91
8	86	88	88	89
16	85	87	87	89

Parameter: Encryption Time (mS) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC
1	1660	2447	1227	1820	1041	1578
2	3237	4790	2377	3555	2048	3069
4	6494	9553	4788	7173	4099	6150
8	13689	20179	10077	15130	8644	12954
16	27426	40415	20199	30326	17322	25981

Parameter: Decryption Time (mS) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC
1	1616	2330	1205	1789	1041	1544
2	3193	4519	2352	3519	2011	3040
4	6490	9209	4776	7171	4102	6156
8	13645	19375	10066	15082	8623	12925
16	27399	38936	20193	30294	17301	25954

Parameter: Encryption Power (mW) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC
1	554	817	421	613	357	552
2	1102	1621	806	1197	682	1042
4	2171	3208	1612	2410	1372	2073
8	4569	6731	3364	5042	2893	4325
16	9156	13510	6735	10134	5788	8670

Parameter: Decryption Power (mW) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC
1	562	815	411	602	354	534
2	1081	1506	802	1178	673	1038
4	2175	3072	1608	2417	1375	2066
8	4548	6463	3357	5035	2874	4330
16	9148	12995	6743	10109	5772	8674

Parameter: Security (%) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC
1	89	93	92	94	94	94
2	88	89	89	92	91	93
4	87	88	88	90	90	91
8	85	88	87	90	89	90
16	85	86	87	89	88	90

Parameter: Encryption Time (mS) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC
1	1654	2441	1204	1728	1046	1520
2	3233	4748	2378	3362	2050	2984
4	6490	9559	4775	6753	4096	6008
8	13670	20174	10082	14252	8652	12693
16	27432	40443	20214	28603	17318	25402

Parameter: Decryption Time (mS) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC
1	1616	2322	1205	1708	1027	1517
2	3184	4530	2366	3336	2026	2967
4	6496	9212	4778	6755	4099	6017
8	13657	19396	10050	14240	8616	12647
16	27401	38958	20210	28571	17310	25388

Parameter: Encryption Power (mW) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC
1	571	840	421	592	348	513
2	1100	1582	796	1146	699	1003
4	2179	3228	1604	2263	1381	2029
8	4577	6750	3361	4765	2893	4246

16	9165	13496	6738	9559	5775	8483

Parameter: Decryption Power (mW) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC
1	541	785	415	584	350	510
2	1086	1536	805	1140	693	1010
4	2192	3088	1604	2275	1368	2008
8	4579	6496	3368	4761	2877	4220
16	9133	12997	6742	9529	5778	8472

Parameter: Security (%) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC
1	91	92	92	94	92	96
2	88	89	89	92	91	92
4	86	89	88	91	90	91
8	86	88	88	89	89	91
16	85	87	87	89	87	90

Parameter: Encryption Time (mS) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC
1	1660	2447	1227	1820	1041	1578	1015	1305
2	3237	4790	2377	3555	2048	3069	2001	2547
4	6494	9553	4788	7173	4099	6150	3997	5122
8	13689	20179	10077	15130	8644	12954	8424	10813
16	27426	40415	20199	30326	17322	25981	16899	21666

Parameter: Decryption Time (mS) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC
1	1616	2330	1205	1789	1041	1544	1014	1282
2	3193	4519	2352	3519	2011	3040	1974	2533
4	6490	9209	4776	7171	4102	6156	4000	5125
8	13645	19375	10066	15082	8623	12925	8406	10785
16	27399	38936	20193	30294	17301	25954	16877	21646

Parameter: Encryption Power (mW) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-	SSO-AC-ECC

							RSA	
1	554	817	421	613	357	552	341	424
2	1102	1621	806	1197	682	1042	673	792
4	2171	3208	1612	2410	1372	2073	1337	1598
8	4569	6731	3364	5042	2893	4325	2820	3374
16	9156	13510	6735	10134	5788	8670	5633	6752

Parameter: Decryption Power (mW) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC
1	562	815	411	602	354	534	339	399
2	1081	1506	802	1178	673	1038	669	799
4	2175	3072	1608	2417	1375	2066	1344	1600
8	4548	6463	3357	5035	2874	4330	2815	3361
16	9148	12995	6743	10109	5772	8674	5634	6739

Parameter: Security (%) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC
1	89	93	92	94	94	94	92	95
2	88	89	89	92	91	93	91	92
4	87	88	88	90	90	91	90	92
8	85	88	87	90	89	90	89	90
16	85	86	87	89	88	90	88	89

Parameter: Encryption Time (mS) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC
1	1654	2441	1204	1728	1046	1520	1009	1299
2	3233	4748	2378	3362	2050	2984	1991	2550
4	6490	9559	4775	6753	4096	6008	3995	5119
8	13670	20174	10082	14252	8652	12693	8427	10819
16	27432	40443	20214	28603	17318	25402	16893	21646

Parameter: Decryption Time (mS) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC
1	1616	2322	1205	1708	1027	1517	995	1290
2	3184	4530	2366	3336	2026	2967	1967	2515
4	6496	9212	4778	6755	4099	6017	3990	5127
8	13657	19396	10050	14240	8616	12647	8409	10779
16	27401	38958	20210	28571	17310	25388	16887	21643

Parameter: Encryption Power (mW) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC
1	571	840	421	592	348	513	345	406
2	1100	1582	796	1146	699	1003	669	812
4	2179	3228	1604	2263	1381	2029	1341	1597
8	4577	6750	3361	4765	2893	4246	2824	3374
16	9165	13496	6738	9559	5775	8483	5646	6739

Parameter: Decryption Power (mW) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC
1	541	785	415	584	350	510	342	420
2	1086	1536	805	1140	693	1010	655	784
4	2192	3088	1604	2275	1368	2008	1344	1608
8	4579	6496	3368	4761	2877	4220	2819	3360
16	9133	12997	6742	9529	5778	8472	5641	6739

Parameter: Security (%) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC
1	91	92	92	94	92	96	93	94
2	88	89	89	92	91	92	92	93
4	86	89	88	91	90	91	90	91
8	86	88	88	89	89	91	89	91
16	85	87	87	89	87	90	89	89

Parameter: Encryption Time (mS) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC	BFO-AC-RSA	BFO-AC-ECC
1	1660	2447	1227	1820	1041	1578	1015	1305	996	1219
2	3237	4790	2377	3555	2048	3069	2001	2547	1944	2412
4	6494	9553	4788	7173	4099	6150	3997	5122	3895	4849
8	13689	20179	10077	15130	8644	12954	8424	10813	8214	10241
16	27426	40415	20199	30326	17322	25981	16899	21666	16469	20503

Parameter: Decryption Time (mS) [Android]

Data Size	RSA	ECC	AC-RSA	AC-ECC	PSO-	PSO-	SSO-	SSO-	BFO-	BFO-

(MB)					AC-RSA	AC-ECC	AC-RSA	AC-ECC	AC-RSA	AC-ECC
1	1616	2330	1205	1789	1041	1544	1014	1282	972	1224
2	3193	4519	2352	3519	2011	3040	1974	2533	1911	2398
4	6490	9209	4776	7171	4102	6156	4000	5125	3897	4850
8	13645	19375	10066	15082	8623	12925	8406	10785	8185	10208
16	27399	38936	20193	30294	17301	25954	16877	21646	16451	20479

Parameter: Encryption Power (mW) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC	BFO-AC-RSA	BFO-AC-ECC
1	554	817	421	613	357	552	341	424	341	388
2	1102	1621	806	1197	682	1042	673	792	664	776
4	2171	3208	1612	2410	1372	2073	1337	1598	1303	1566
8	4569	6731	3364	5042	2893	4325	2820	3374	2739	3283
16	9156	13510	6735	10134	5788	8670	5633	6752	5494	6557

Parameter: Decryption Power (mW) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC	BFO-AC-RSA	BFO-AC-ECC
1	562	815	411	602	354	534	339	399	334	397
2	1081	1506	802	1178	673	1038	669	799	649	783
4	2175	3072	1608	2417	1375	2066	1344	1600	1299	1561
8	4548	6463	3357	5035	2874	4330	2815	3361	2741	3259
16	9148	12995	6743	10109	5772	8674	5634	6739	5494	6544

Parameter: Security (%) [Android]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC	BFO-AC-RSA	BFO-AC-ECC
1	89	93	92	94	94	94	92	95	94	96
2	88	89	89	92	91	93	91	92	91	94
4	87	88	88	90	90	91	90	92	90	92
8	85	88	87	90	89	90	89	90	90	91
16	85	86	87	89	88	90	88	89	89	91

Parameter: Encryption Time (mS) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC	BFO-AC-RSA	BFO-AC-ECC
1	1654	2441	1204	1728	1046	1520	1009	1299	986	1235
2	3233	4748	2378	3362	2050	2984	1991	2550	1935	2431
4	6490	9559	4775	6753	4096	6008	3995	5119	3888	4846

8	13670	20174	10082	14252	8652	12693	8427	10819	8209	10228
16	27432	40443	20214	28603	17318	25402	16893	21646	16451	20489

Parameter: Decryption Time (mS) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC	BFO-AC-RSA	BFO-AC-ECC
1	1616	2322	1205	1708	1027	1517	995	1290	988	1226
2	3184	4530	2366	3336	2026	2967	1967	2515	1914	2384
4	6496	9212	4778	6755	4099	6017	3990	5127	3896	4846
8	13657	19396	10050	14240	8616	12647	8409	10779	8187	10192
16	27401	38958	20210	28571	17310	25388	16887	21643	16454	20474

Parameter: Encryption Power (mW) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC	BFO-AC-RSA	BFO-AC-ECC
1	571	840	421	592	348	513	345	406	337	410
2	1100	1582	796	1146	699	1003	669	812	662	779
4	2179	3228	1604	2263	1381	2029	1341	1597	1303	1549
8	4577	6750	3361	4765	2893	4246	2824	3374	2745	3270
16	9165	13496	6738	9559	5775	8483	5646	6739	5494	6541

Parameter: Decryption Power (mW) [Windows]

Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC	BFO-AC-RSA	BFO-AC-ECC
1	541	785	415	584	350	510	342	420	332	410
2	1086	1536	805	1140	693	1010	655	784	641	765
4	2192	3088	1604	2275	1368	2008	1344	1608	1312	1558
8	4579	6496	3368	4761	2877	4220	2819	3360	2741	3253
16	9133	12997	6742	9529	5778	8472	5641	6739	5491	6540

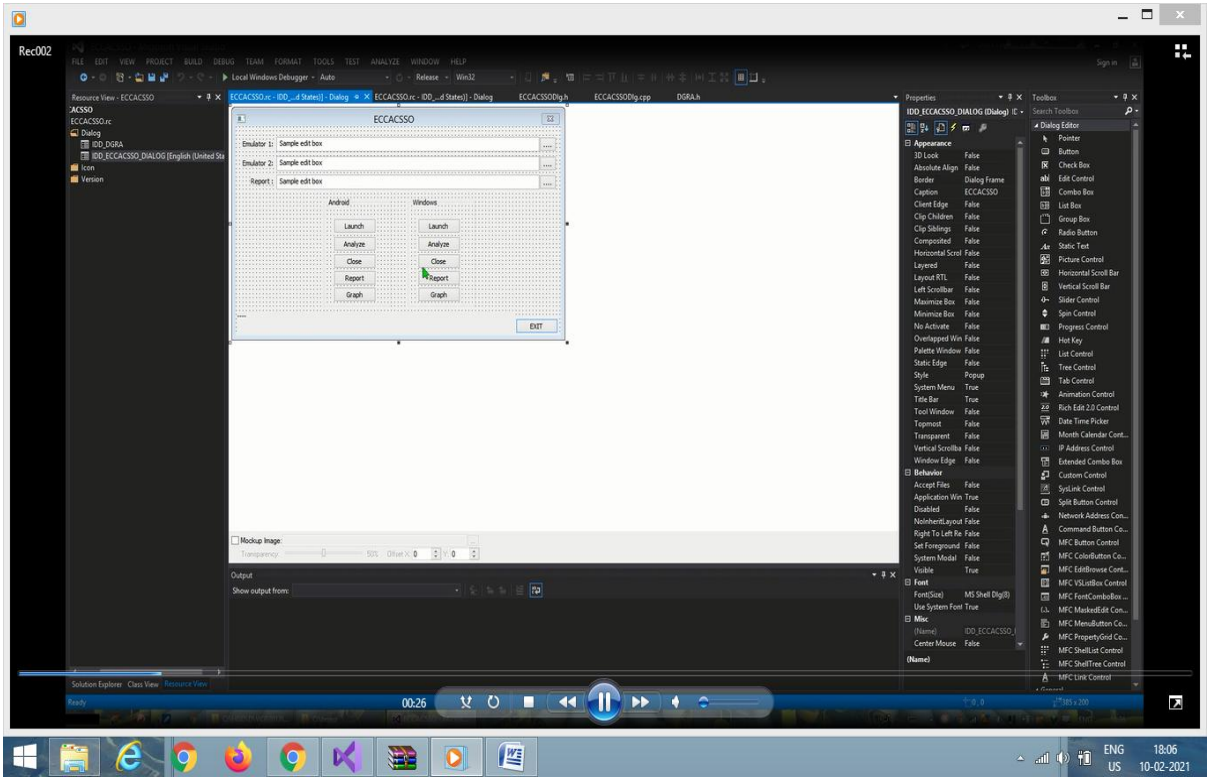
Data Size (MB)	RSA	ECC	AC-RSA	AC-ECC	PSO-AC-RSA	PSO-AC-ECC	SSO-AC-RSA	SSO-AC-ECC	BFO-AC-RSA	BFO-AC-ECC
1	91	92	92	94	92	96	93	94	95	96
2	88	89	89	92	91	92	92	93	91	94
4	86	89	88	91	90	91	90	91	91	92
8	86	88	88	89	89	91	89	91	89	91
16	85	87	87	89	87	90	89	89	88	91

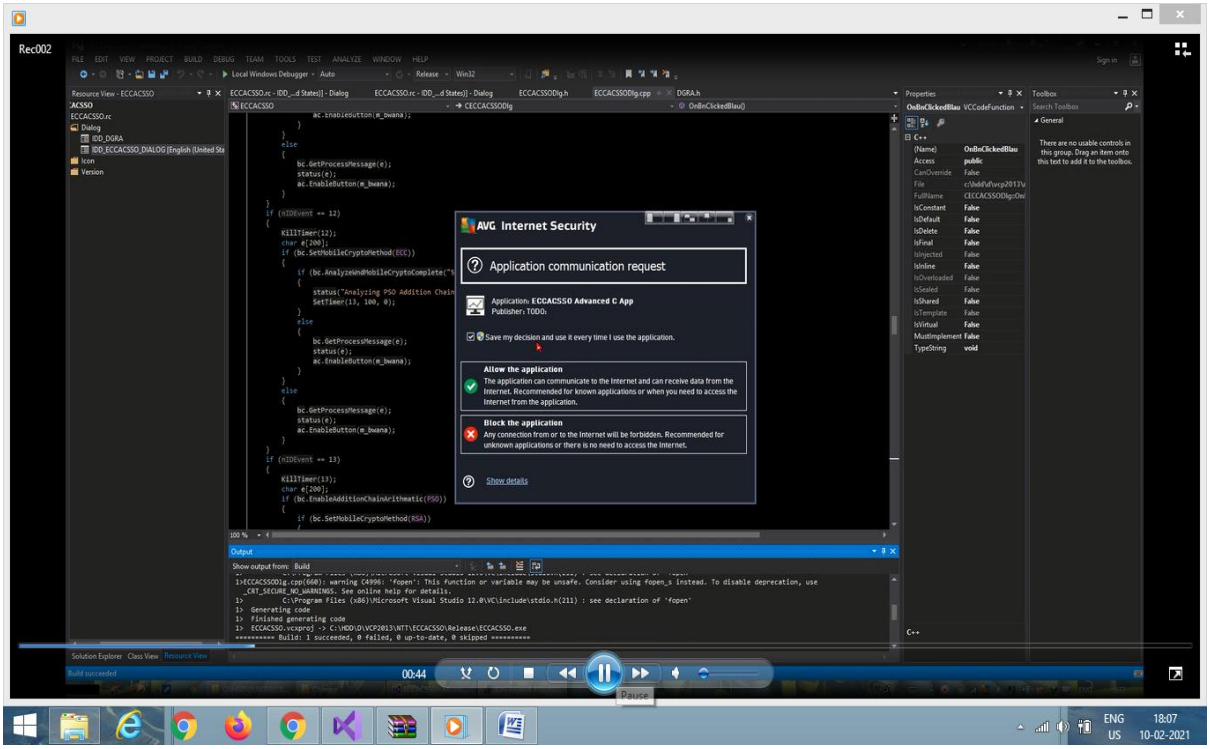
---

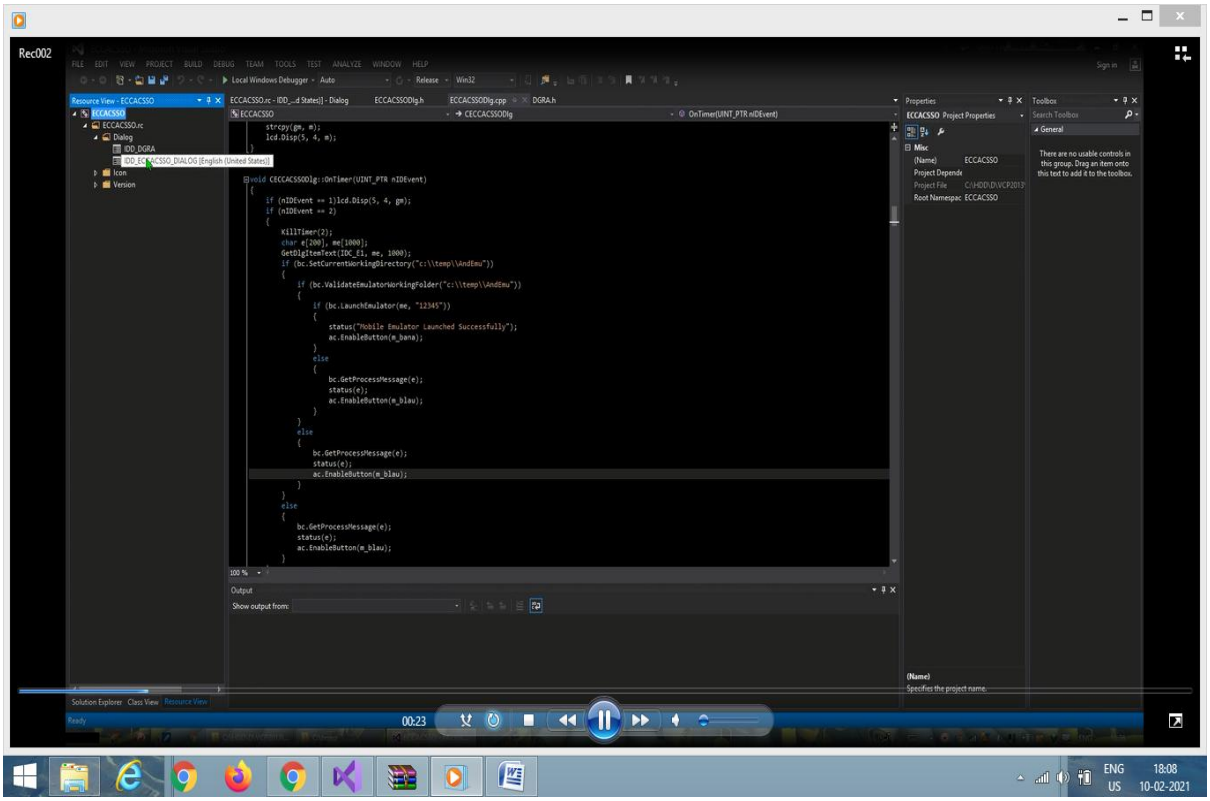
end of report

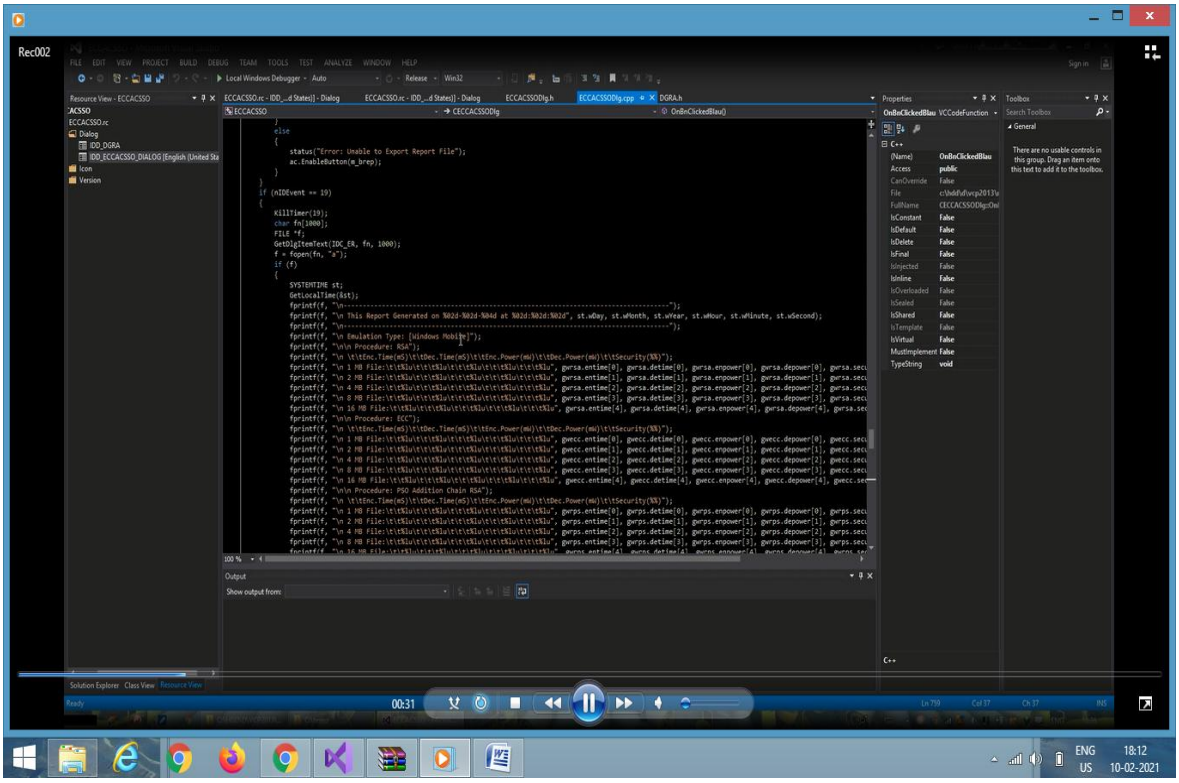
APPENDIX - A.5

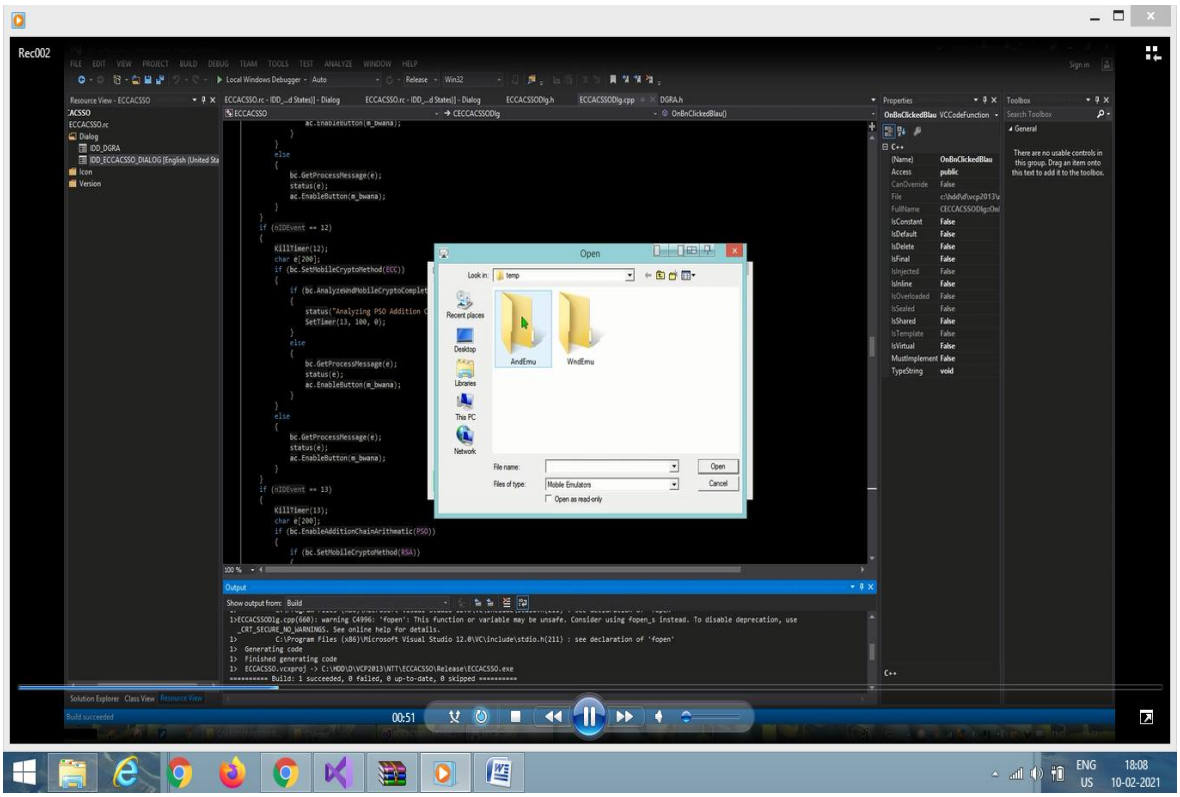
SAMPLE SCREENSHOTS

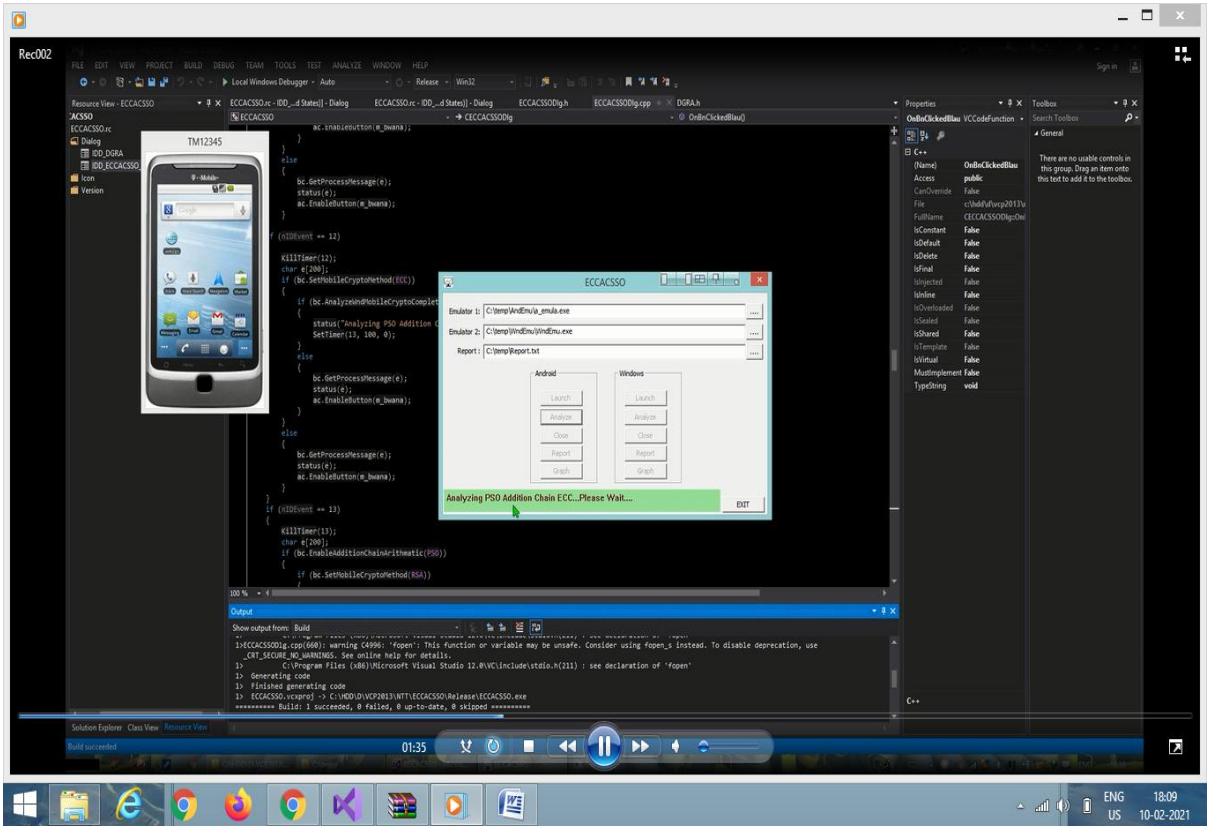


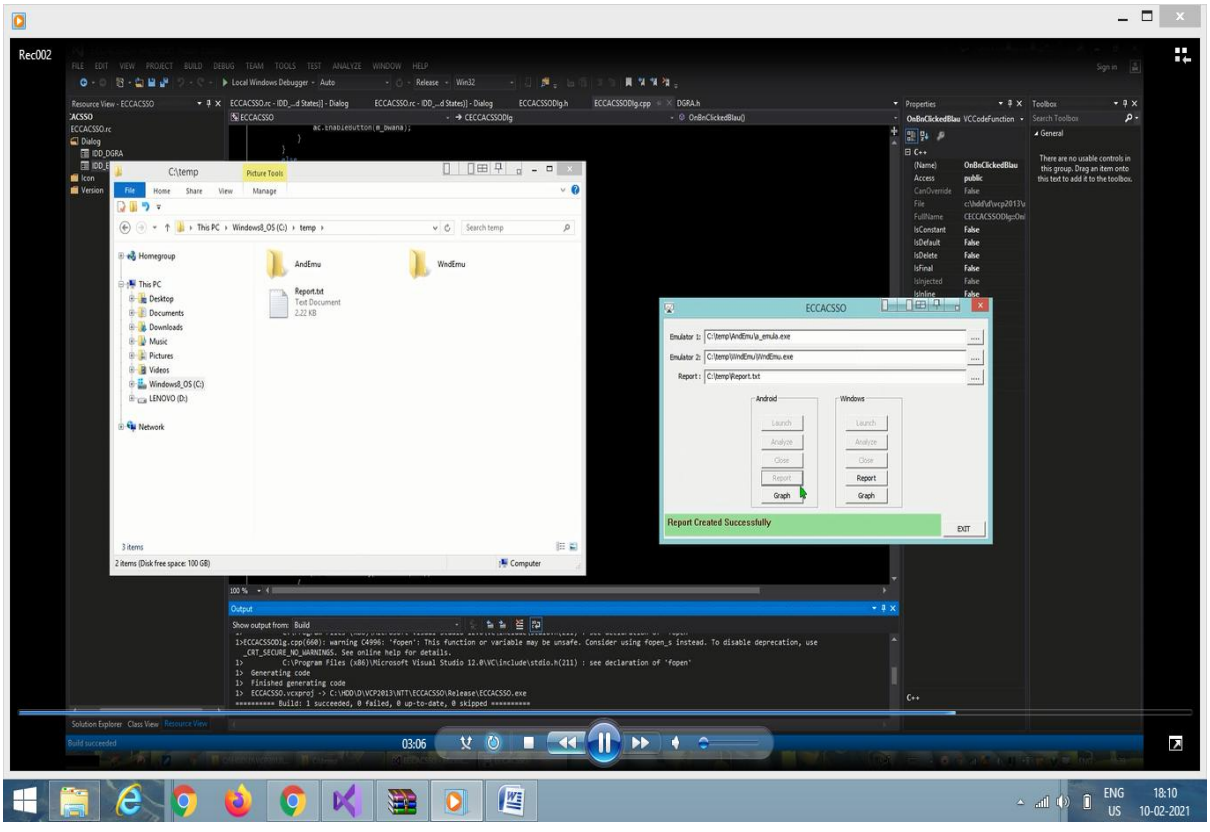


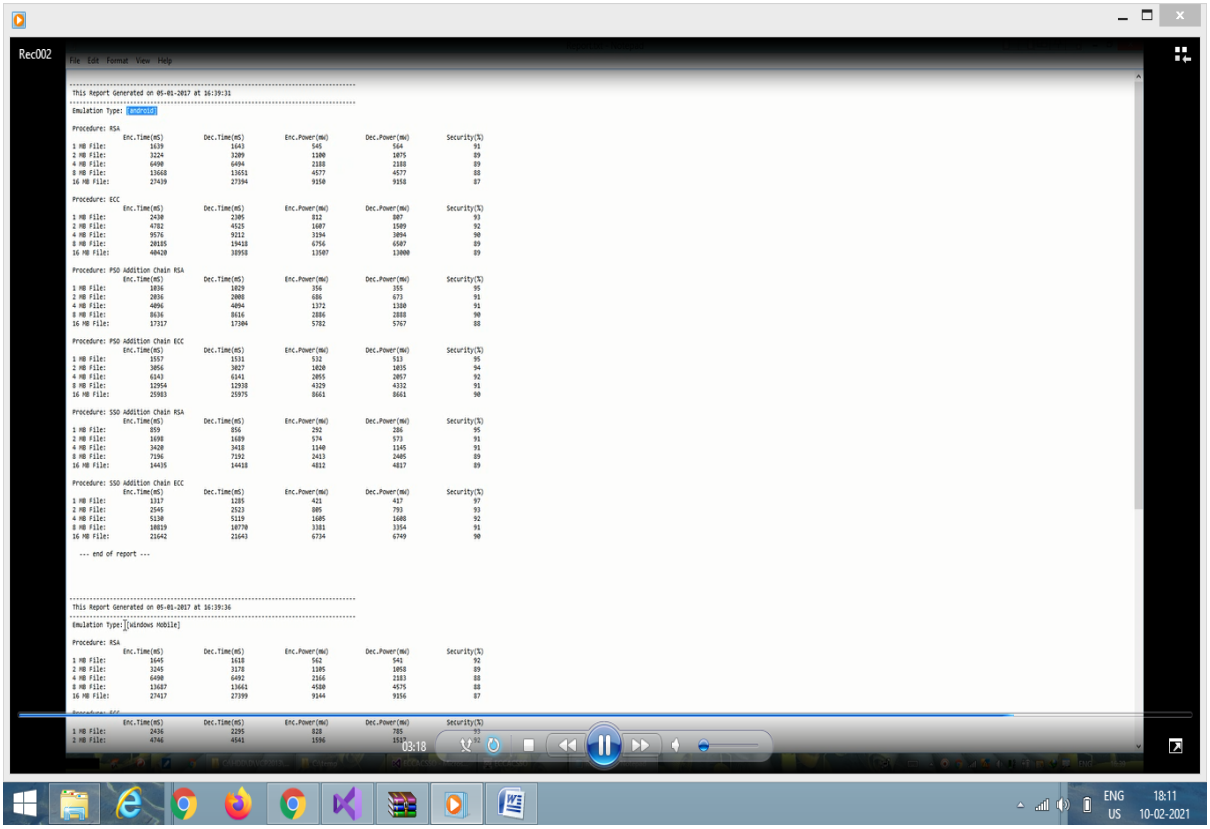

Rec002
ECCACSSO - Microsoft Visual Studio
00:23
ENG
US
18:08
10-02-2021

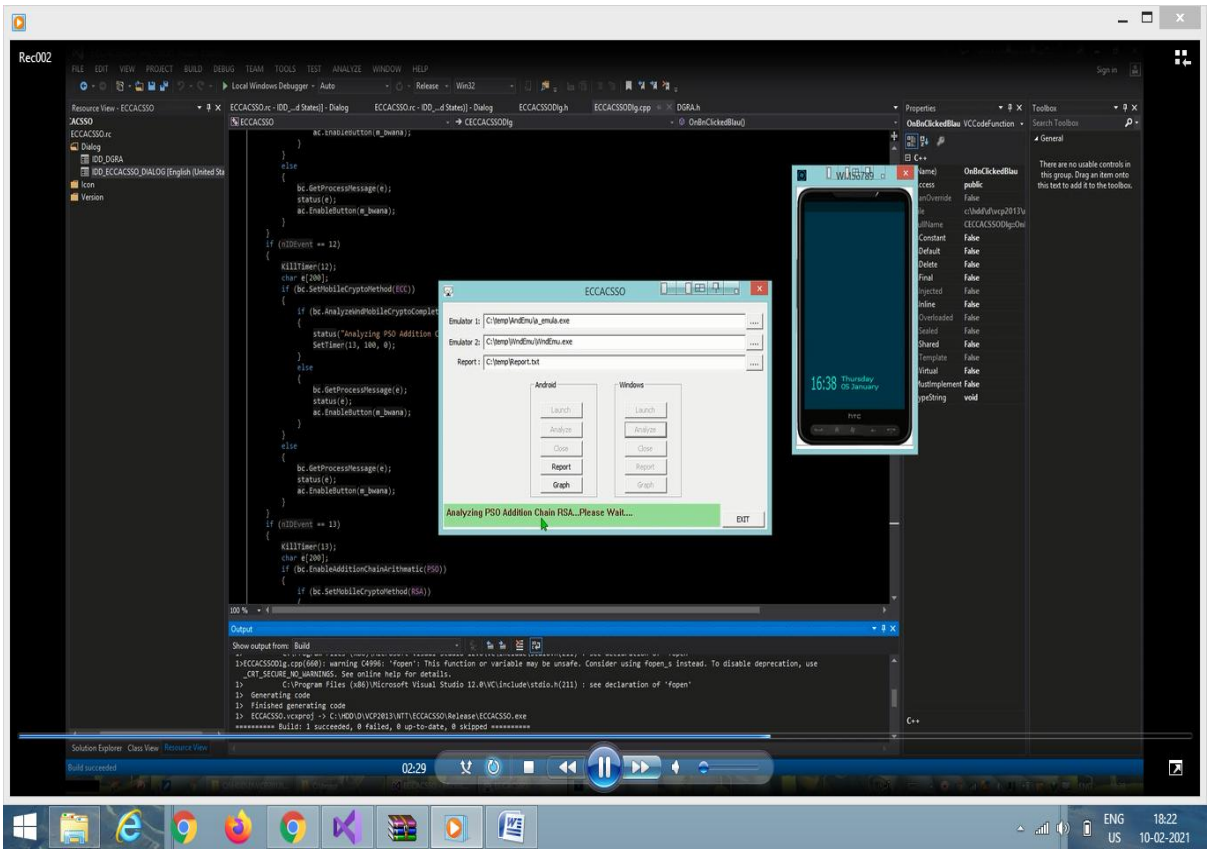

Rec002
00:31
ENG
US
18:12
10-02-2021

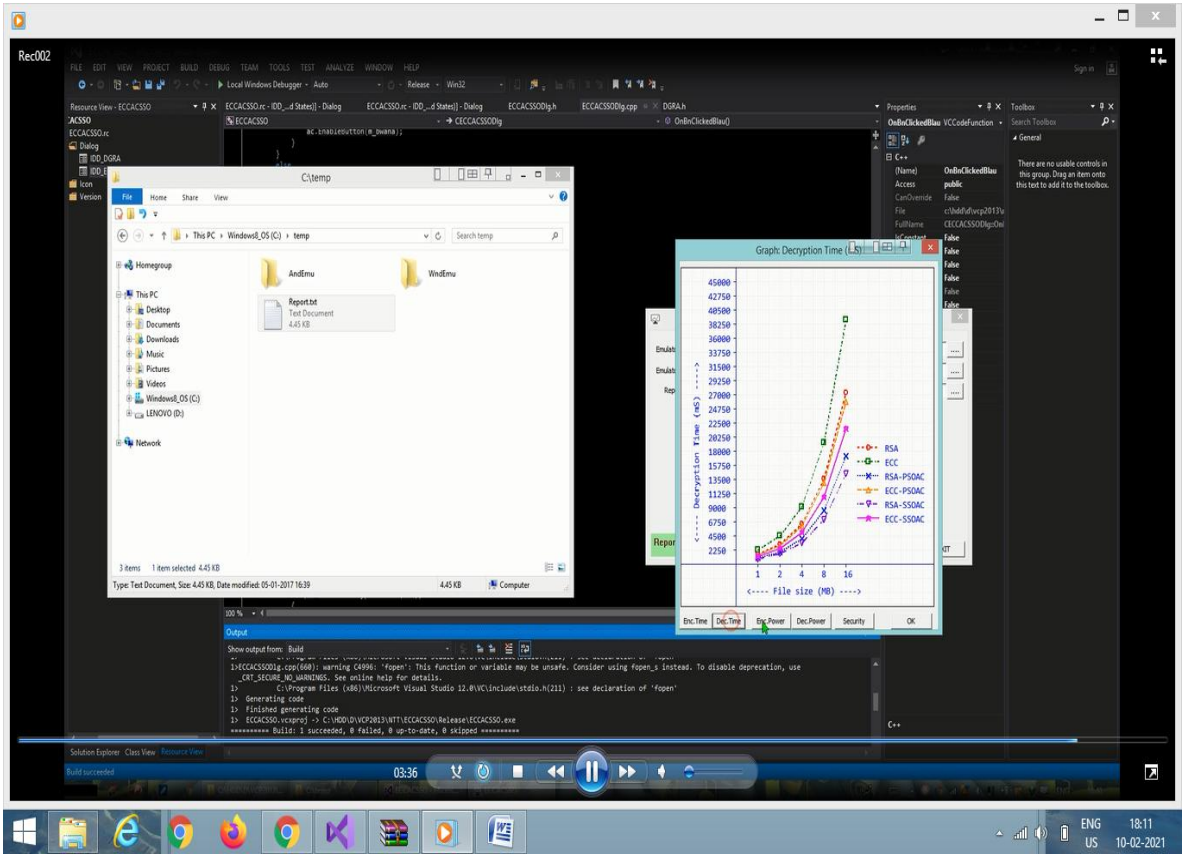

Rec002
Open
00:51
ENG
US
18:08
10-02-2021

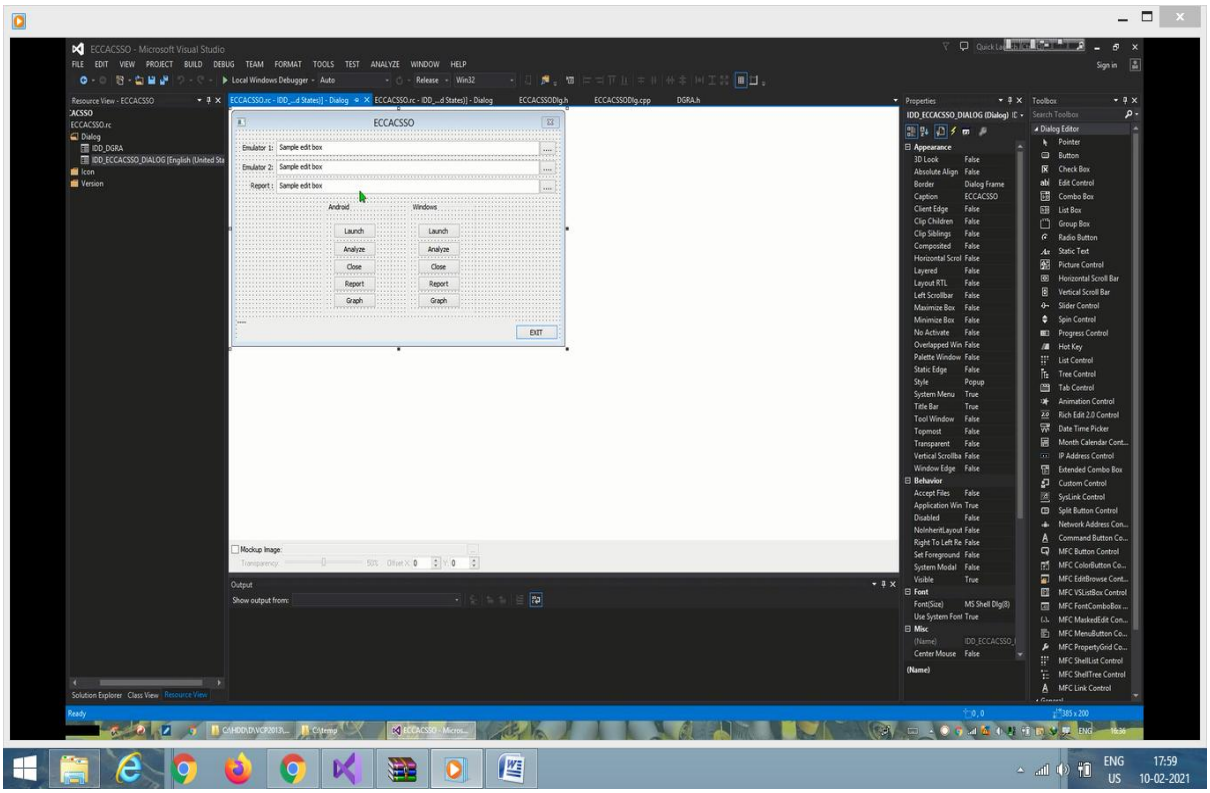

Rec002
TM12345
ECCACSSO
Emulator 1: C:\temp\AndEmu\a_emula.exe
Emulator 2: C:\temp\WindEmu\WindEmu.exe
Report : C:\temp\Report.txt
Android
Windows
Launch
Analyze
Close
Report
Graph
Analyzing PSO Addition Chain ECC...Please Wait....
EXIT
01:35
ENG
US
18:09
10-02-2021


Rec002
C:\temp
AndEmu
WindEmu
Report.txt
Text Document
2.22 KB
ECCACSSO
Emulator 1: C:\temp\AndEmu\a_emula.exe
Emulator 2: C:\temp\WindEmu\WindEmu.exe
Report : C:\temp\Report.txt
Android
Windows
Launch
Analyze
Close
Report
Graph
Report Created Successfully
EXIT
03:06
ENG
US
18:10
10-02-2021


Rec002
This Report Generated on 05-01-2017 at 16:39:31
Emulation Type: android
Procedure: RSA
Enc.Time(mS) Dec.Time(mS) Enc.Power(mW) Dec.Power(mW) Security(%)
1 MB File: 1639 1643 565 564 91
2 MB File: 3224 3209 1100 1075 89
4 MB File: 6498 6494 2188 2188 89
8 MB File: 13668 13651 4577 4577 88
16 MB File: 27439 27394 9150 9158 87
Procedure: ECC
Enc.Time(mS) Dec.Time(mS) Enc.Power(mW) Dec.Power(mW) Security(%)
1 MB File: 2438 2385 812 807 93
2 MB File: 4782 4525 1607 1589 92
4 MB File: 9576 9212 3194 3094 90
8 MB File: 20185 19418 6756 6587 89
16 MB File: 40420 38958 13507 13000 89
Procedure: PSO Addition Chain RSA
Enc.Time(mS) Dec.Time(mS) Enc.Power(mW) Dec.Power(mW) Security(%)
1 MB File: 1836 1829 356 355 95
2 MB File: 2036 2008 686 673 91
4 MB File: 4096 4094 1372 1380 91
8 MB File: 8636 8616 2886 2888 90
16 MB File: 17317 17304 5782 5767 88
Procedure: PSO Addition Chain ECC
Enc.Time(mS) Dec.Time(mS) Enc.Power(mW) Dec.Power(mW) Security(%)
1 MB File: 1567 1531 532 513 95
2 MB File: 3056 3027 1020 1035 94
4 MB File: 6143 6141 2055 2057 92
8 MB File: 12954 12938 4329 4332 91
16 MB File: 25983 25975 8661 8661 90
Procedure: SSO Addition Chain RSA
Enc.Time(mS) Dec.Time(mS) Enc.Power(mW) Dec.Power(mW) Security(%)
1 MB File: 859 856 292 286 95
2 MB File: 1698 1689 574 573 91
4 MB File: 3428 3418 1140 1145 91
8 MB File: 7196 7192 2413 2405 89
16 MB File: 14435 14418 4812 4817 89
Procedure: SSO Addition Chain ECC
Enc.Time(mS) Dec.Time(mS) Enc.Power(mW) Dec.Power(mW) Security(%)
1 MB File: 1317 1285 421 417 97
2 MB File: 2545 2523 805 793 93
4 MB File: 5138 5119 1605 1608 92
8 MB File: 10819 10770 3381 3354 91
16 MB File: 21642 21643 6734 6749 90
--- end of report ---
This Report Generated on 05-01-2017 at 16:39:36
Emulation Type: [windows Mobile]
Procedure: RSA
Enc.Time(mS) Dec.Time(mS) Enc.Power(mW) Dec.Power(mW) Security(%)
1 MB File: 1645 1618 562 541 92
2 MB File: 3245 3178 1105 1058 89
4 MB File: 6498 6492 2166 2183 88
8 MB File: 13687 13661 4580 4575 88
16 MB File: 27417 27399 9144 9156 87
Enc.Time(mS) Dec.Time(mS) Enc.Power(mW) Dec.Power(mW) Security(%)
1 MB File: 2436 2295 828 785 93
2 MB File: 4746 4641 1596 1511 92
03:18
ENG
US
18:11
10-02-2021


Rec002
ECCACSSO
Emulator 1: C:\temp\AndEmu\ia_emula.exe
Emulator 2: C:\temp\WndEmu\WndEmu.exe
Report : C:\temp\Report.txt
Android
Windows
Launch
Analyze
Close
Report
Graph
Analyzing PSO Addition Chain RSA...Please Wait....
EXIT
Output
02:29
ENG
US
18:22
10-02-2021


Rec002
C:\temp
Graph: Decryption Time (LS)
RSA
ECC
RSA-PSOAC
ECC-PSOAC
RSA-SSOAC
ECC-SSOAC
<---- Decryption Time (mS) ---->
<---- File size (MB) ---->
03:36
ENG
US
18:11
10-02-2021


ECCACSSO - Microsoft Visual Studio
ECCACSSO
Emulator 1:
Emulator 2:
Report :
Android
Windows
Launch
Analyze
Close
Report
Graph
EXIT
ENG
US
17:59
10-02-2021