



*Nature has offered many intelligent sensors;  
Only need is to recognize them.*

# **An Empirical Study of Elliptic Curve Cryptography for the Resource Constrained Wireless Sensor Network**

**PRITAM GAJKUMAR SHAH**

**PhD UC Australia**

© 2019 By Pritam Gajkumar Shah, [www.ausjournal.com](http://www.ausjournal.com)

## Abstract

---

Development in Micro Electro Mechanical Systems (MEMS), Very Large Scale Integration (VLSI) and Wireless Communication has opened a new domain in networking arena called Wireless Sensor Networks (WSN). WSN is a special case of a Mobile Ad Hoc Networks (MANET) in which information is gathered, processed and communicated with the help of tiny wireless nodes that are deployed in the field where ordinary networks are unreachable for various environmental and strategic reasons. Wireless sensor nodes uses radio frequencies as a communications medium, and are vulnerable to active and passive attacks from adversaries including node capturing, frequency jamming, and a Denial of Service (DOS) attacks. Most of the research is going on to make WSN secure with Symmetric Key protocols but at the same time Public Key Cryptography (PKC) has received very little attention from researchers.

PKC is based on hardness of a mathematical problem like ‘factoring’ the product of two large primes which is used in *Rivest, Shamir and Adleman* (RSA) algorithm or computing ‘discreet logarithm’ used in *Elliptical Curve Cryptography* (ECC). As compared to RSA, ECC offers same security level but with significantly smaller key size and is the potential candidate in the near future for WSN security. For example, a 160-bit ECC key provides the same level of security as a 1024-bit RSA key, and a 224-bit ECC key provides the same security as a 2048-bit RSA key. Smaller keys in WSN mean faster computation, lower power consumption, and memory and bandwidth savings of sensor node.

Detailed analysis and mathematical modeling of ECC have been investigated in this thesis with special reference to the WSN. Use of *mixed and projective coordinate system*, recoding of integer with *One’s Complement Subtraction* (OCS) method, *splitting of integer* to avoid

Special Power Analysis (SPA) attack, use of *Elastic Window method* to avoid node failure and use of *Hidden generator point* to avoid *man in the middle* attack for WSN have been proposed. These five innovative, novel and industrial applicable algorithms will remarkably improve performance of scalar multiplication process on WSN and will achieve node authenticity, data integrity, confidentiality, and freshness on 8 bit microcontroller of sensor node having limited resources and computational power. They will also provide better flexibility to the node and clean security interface in the WSN architecture. To validate our claims, simulation results obtained on MIRACL crypto library and MATLAB have been provided wherever necessary.

# Table of Contents

---

|  |                              |
|--|------------------------------|
| CERTIFICATE OF AUTHORSHIP.....                                 | ERROR! BOOKMARK NOT DEFINED. |
| ABSTRACT.....  | IV                           |
| LIST OF FIGURES.....   | XI                           |
| LIST OF ALGORITHMS.....  | XIII                         |
| LIST OF TABLES .....   | XIV                          |
| ABBREVIATIONS .....  | XV                           |
| ACKNOWLEDGEMENTS .....   | XVII                         |
| CHAPTER 1 INTRODUCTION.....                                    | 1                            |
| 1.1 BACKGROUND AND MOTIVATION .....                            | 1                            |
| 1.2 RESEARCH QUESTIONS .....                                   | 4                            |
| 1.3 RESEARCH METHODOLOGY .....                                 | 6                            |
| 1.4 RESEARCH CONTRIBUTION .....                                | 7                            |
| 1.5 THESIS OUTLINE.....  | 9                            |
| 1.6 SUMMARY .....  | 11                           |
| CHAPTER 2 WIRELESS SENSOR NETWORKS AND SECURITY: A REVIEW..... | 14                           |
| 2.1 INTRODUCTION: .....  | 14                           |
| 2.2 WSN ARCHITECTURE .....                                     | 16                           |
| 2.2.1 WSN Node .....   | 16                           |
| 2.2.2 Node Power Consumption.....                              | 22                           |
| a) Power required for Communication:.....                      | 22                           |
| b) Power required for data processing:.....                    | 23                           |
| 2.3 WSN PROTOCOL STACK .....                                   | 23                           |
| 2.4 WSN TOPOLOGY .....   | 25                           |
| 2.5 NETWORK LAYER PROTOCOLS EVALUATION .....                   | 27                           |
| 2.6 POSSIBLE ATTACKS ON WSN ILLUSTRATED LAYER WISE .....       | 28                           |
| 2.6.1. Physical Layer.....                                     | 28                           |
| a. Denial of Service (DOS) attack: .....                       | 28                           |
| b. Frequency Jamming: .....                                    | 28                           |
| c. Radio Interferences: .....                                  | 29                           |
| d. Tampering Node: .....                                       | 29                           |

|  |           |
|--|-----------|
| <i>d. Simple Power Analysis Attack:</i> .....                                      | 29        |
| 2.6.2 DATA LINK LAYER .....  | 31        |
| <i>a. Exhaustion:</i> .....  | 31        |
| <i>b. Interrogation:</i> .....   | 31        |
| <i>c. Sybil attack:</i> .....  | 31        |
| 2.6.3 NETWORK LAYER.....   | 31        |
| <i>a. Sinkhole:</i> .....  | 31        |
| <i>b. Hello Flood:</i> .....   | 32        |
| <i>c. Selective Forwarding:</i> .....  | 32        |
| <i>d. Wormhole Attacks:</i> .....  | 32        |
| <i>e. Replayed Routing Information:</i> .....                                      | 32        |
| <i>g. Misguiding path</i> .....  | 33        |
| <i>i. Homing:</i> .....  | 33        |
| 2.6.4 TRANSMISSION LAYER .....   | 33        |
| <i>De-synchronization Attacks:</i> .....   | 33        |
| 2.6.5 APPLICATION LAYER .....  | 34        |
| <i>a. Overwhelm attack:</i> .....  | 34        |
| <i>b. Path-based DOS attack:</i> .....   | 34        |
| <i>c. Deluge (reprogram) attack:</i> .....   | 34        |
| 2.7 A SURVEY OF CRYPTOGRAPHY PROTOCOLS FOR WSN.....                                | 34        |
| 2.7.1 SPINS: Security Protocol for Sensor Network .....                            | 35        |
| <i>a. SNEP:</i> .....  | 36        |
| <i>b. <math>\mu</math>TESLA :</i> .....  | 36        |
| 2.7.2 TinySec: .....   | 37        |
| 2.8. SUMMARY .....   | 38        |
| <b>CHAPTER 3 ELLIPTICAL CURVE CRYPTOGRAPHY MODELLING .....</b>                     | <b>40</b> |
| 3.1 INTRODUCTION .....   | 40        |
| 3.2 DEFINITION OF ELLIPTICAL CURVE .....   | 40        |
| 3.2 PYRAMID OF ELLIPTICAL CURVE CRYPTOGRAPHY (ECC).....                            | 44        |
| 3.3 ELLIPTIC CURVE PROTOCOLS FOR WSN .....   | 45        |
| 3.3.1 <i>Elliptic curve key generation</i> .....                                   | 45        |
| 3.3.2 <i>Definition of Elliptic Curve Discrete Logarithm Problem (ECDLP)</i> ..... | 46        |
| 3.3.3 <i>Elliptical Curve Diffie-Hellman Protocol (ECDH)</i> .....                 | 47        |
| 3.3.4 <i>Comparison between RSA and ECC for WSN</i> .....                          | 50        |
| 3.3.5 ELGAMAL ELLIPTIC CURVE PROTOCOL .....  | 52        |
| 3.4 POINT ADDITION AND POINT DOUBLING ON ELLIPTICAL CURVE .....                    | 56        |
| 3.5 FINITE FIELD ARITHMETIC.....   | 58        |
| <i>a. Definition of Modular Reduction:</i> .....                                   | 59        |
| <i>b. Definition of Modular Inversion:</i> .....                                   | 59        |
| 3.6 FIELDS OF ODD CHARACTERISTIC: .....  | 59        |
| 3.6.1 <i>Moduli of Special form or pre computed moduli</i> .....                   | 60        |

|  |            |
|--|------------|
| 3.6.2 Residue number system arithmetic.....  | 60         |
| 3.6.3 Barrett Reduction.....   | 61         |
| 3.6.4 Montgomery Reduction .....   | 62         |
| 3.7 FIELDS OF CHARACTERISTIC TWO.....  | 66         |
| 3.7.1 Polynomial Bases: .....  | 66         |
| a. Modular reduction .....   | 66         |
| b. Multiplication.....   | 68         |
| c. Inversion .....   | 68         |
| 3.7.2 Normal bases .....   | 69         |
| 3.7.3 Subfield bases .....   | 72         |
| 3.8 SUMMARY .....  | 74         |
| <b>CHAPTER 4 OPTIMIZATION OF COORDINATE SYSTEM FOR ECC ON WSN .....</b>  | <b>75</b>  |
| 4.1 INTRODUCTION .....   | 75         |
| 4.2 COORDINATE SYSTEMS IN ELLIPTICAL CURVE.....  | 77         |
| 4.2.1 Affine coordinates.....  | 77         |
| 4.2.2 Projective Coordinate.....   | 78         |
| 4.2.3 Jacobian and Chudnovsky Jacobian Coordinates.....  | 80         |
| 4.2.4 Modified Jacobian Coordinates.....   | 82         |
| 4.2.5 Mixed Coordinates .....  | 82         |
| 4.3 IMPLEMENTATION ON MIRACL CRYPTO LIBRARY .....  | 87         |
| 4.4 SUMMARY .....  | 89         |
| <b>CHAPTER 5 ONE’S COMPLEMENT SUBTRACTION (OCS) ALGORITHM PROPOSED FOR RECODING OF INTEGER ON WSN PLATFORM .....</b> | <b>91</b>  |
| 5.1 INTRODUCTION .....   | 91         |
| 5.2 ECDH PROTOCOL IMPLEMENTED ON WSN TO ACHIEVE DATA CONFIDENTIALITY .....   | 92         |
| 5.3 THE EXISTING METHODS OF INTEGER RECODING .....   | 94         |
| 5.3.1 Binary Method .....  | 94         |
| 5.3.2 Non Adjacent Form (NAF) method.....  | 96         |
| 5.3.4 Mutual Opposite Form (MOF) Method .....  | 99         |
| 5.6 PROPOSED ONE’S COMPLEMENT SUBTRACTION (OCS) ALGORITHM FOR RECODING OF SCALAR $k$ .....                           | 100        |
| 5.6.1 THE 10’S COMPLEMENT.....   | 100        |
| 5.6.2 THE 9’S COMPLEMENT .....   | 102        |
| 5.6.3 BINARY SUBTRACTION BY UTILIZATION OF 2’S COMPLEMENT .....  | 103        |
| 5.6.4 BINARY SUBTRACTION BY UTILIZATION OF 1’S COMPLEMENT .....  | 106        |
| 5.6.5 PROPOSED TCS ALGORITHM.....  | 109        |
| 5.7 PERFORMANCE EVALUATION OF OCS ALGORITHM ON MATLAB.....   | 110        |
| 5.8 SUMMARY .....  | 117        |
| <b>CHAPTER 6 PROPOSED WINDOW OCS ALGORITHM FOR PREVENTION OF SPA IN WSN .....</b>                                    | <b>118</b> |
| 6.1 INTRODUCTION .....   | 118        |



|   |            |
|---|------------|
| 6.2 OVERVIEW OF SIDE CHANNEL ATTACKS ON WSN NODE .....  | 119        |
| 6.2.1 CMOS Inverter Circuit and Simple Power Analysis .....   | 120        |
| 6.2.2 Enhanced Simple Power Analysis .....  | 122        |
| 6.2.3 Differential Power Analysis .....   | 122        |
| 6.2.4 Electro magnetic Analysis .....   | 123        |
| 6.2.5 Fault and Timings Attacks .....   | 124        |
| 6.3 OVERVIEW OF EXISTING SPA COUNTER MEASURES .....   | 124        |
| 6.3.1 Double and Add Always .....   | 125        |
| 6.3.2 Montgomery Ladder .....   | 125        |
| 6.3.3 Identical formulae for point addition and doublings .....   | 126        |
| 6.4 PROPOSED WINDOW OCS METHOD TO AVOID SPA IN WSN .....  | 127        |
| 6.5 SUMMARY .....   | 129        |
| <b>CHAPTER 7 PROPOSED ELASTIC WINDOW METHOD OF SCALAR MULTIPLICATION FOR WSN .....</b>                              | <b>130</b> |
| 7.1 INTRODUCTION .....  | 130        |
| 7.2 OVERVIEW OF EXISTING METHODS OF SCALAR MULTIPLICATION .....   | 131        |
| 7.2.1 THE BINARY METHOD FOR SCALAR MULTIPLICATION .....   | 131        |
| 7.2.2 THE M-ARY METHOD .....  | 132        |
| 7.2.3 SLIDING WINDOW METHOD .....   | 133        |
| 7.3 EFFECT OF WINDOW SIZE ON MEMORY UTILIZATION OF MICA NODES .....   | 139        |
| 7.4 STACK DEPTH ANALYSIS .....  | 140        |
| 7.5 PROPOSED ELASTIC WINDOW ALGORITHM FOR SCALAR MULTIPLICATION .....   | 141        |
| 7.6 SUMMARY .....   | 145        |
| <b>CHAPTER 8 ALGORITHM BASED ON HIDDEN GENERATOR POINT PROPOSED FOR WSN TO AVOID MAN IN THE MIDDLE ATTACK .....</b> | <b>146</b> |
| 8.1 INTRODUCTION .....  | 146        |
| 8.2 PROPOSED NEW ECC PROTOCOL BASED ON HIDDEN GENERATOR POINT .....   | 147        |
| 8.3 MULTI-AGENT SYSTEM IMPLEMENTATION OF ECC PUBLIC KEY .....   | 154        |
| 8.4 SUMMARY .....   | 156        |
| <b>CHAPTER 9 PROPOSED UNI-COORDINATE SYSTEM OF ECC FOR WSN .....</b>  | <b>157</b> |
| 9.1 INTRODUCTION .....  | 157        |
| 9.2 QUADRATIC EQUATIONS IN FIELD OF ODD CHARACTERISTICS: .....  | 157        |
| 9.3 SOLVING QUADRATIC EQUATIONS IN BINARY FIELD .....   | 159        |
| 9.4 SUMMARY .....   | 161        |
| <b>CHAPTER 10 SUMMARY OF THE RESEARCH .....</b>   | <b>162</b> |
| <b>CHAPTER 11 FUTURE SCOPE .....</b>  | <b>165</b> |
| <b>BIBLIOGRAPHY .....</b>   | <b>166</b> |
| <b>APPENDIX A NIST RECOMMENDED CURVES FOR CRYPTOGRAPHY .....</b>  | <b>177</b> |

|  |            |
|--|------------|
| <b>APPENDIX B LIST OF MY PUBLICATIONS AND PATENTS.....</b>           | <b>194</b> |
| <b>APPENDIX C RESULTS OBTAINED ON MIRACL CRYPTO LIBRARY .....</b>    | <b>197</b> |
| <b>APPENDIX D SAMPLE SOURCE CODES WRITTEN IN C, C++ FOR ECC.....</b> | <b>206</b> |

## List of Figures

---

|   |     |
|---|-----|
| Figure 2. 1 A Typical WSN Architecture .....  | 14  |
| Figure 2. 2 Block Diagram of WSN Node.....  | 16  |
| Figure 2. 3 Fleck3B WSN Node by CSIRO [7].....  | 17  |
| Figure 2. 4 Images of MICA Node (on Left) and Telos Node (on Right).....                      | 19  |
| Figure 2. 5 WSN Protocol Stack with Proposed Security Plane .....                             | 24  |
| Figure 2. 6 Deployment of Nodes (Delphi Distribution) .....                                   | 26  |
| Figure 2. 7 Deployment of Nodes (Gaussian Distribution) .....                                 | 26  |
| Figure 2. 8 Deployment of Nodes (Exponential Distribution).....                               | 27  |
| Figure 2. 9 Power Traces of SC140DS Processor.....  | 30  |
| Figure 2. 10 $\mu$ TESLA One Way Key Function[63] .....                                       | 37  |
| Figure 3. 1 An Elliptical Curve Example.....  | 42  |
| Figure 3. 2 An Elliptical Curve and Its Group Elements.....                                   | 43  |
| Figure 3. 3 ECC Operational Pyramid .....   | 45  |
| Figure 3. 4 Diffie Hellman Key Exchange Protocol.....   | 52  |
| Figure 3. 5 Point Addition, Point Doubling Operations on Elliptical Curves .....              | 56  |
| Figure 4. 1 Evaluation of Coordinate System for ECDH and ECDSA on MIRACL Crypto Library ..... | 88  |
| Figure 5. 1 ECDH Protocol Implemented on WSN .....  | 92  |
| Figure 5. 2 MATLAB code for Decimal to Binary Conversion.....                                 | 111 |
| Figure 5. 3 MATLAB Code For Decimal to MOF Form.....  | 112 |
| Figure 5. 4 MATLAB Code For Decimal To NAF Form .....   | 113 |
| Figure 5. 5 MATLAB code for OCS Algorithm.....  | 114 |
| Figure 5. 6 MATLAB code for TCS algorithm.....  | 115 |
| Figure 5. 7 Comparison of OCS with other algorithms on MATLAB .....                           | 116 |
| Figure 5. 8 Trade Off Between Window Size $w$ and Computational Cost.....                     | 138 |
| Figure 5. 9 Trade off between window size $w$ and number of pre-computations .....            | 138 |
| Figure 6. 1 Power traces revealing value of private key of the WSN node [63] .....            | 119 |
| Figure 6. 2 Block diagram of SPA setup .....  | 119 |
| Figure 6. 3 CMOS Logic Inverter Circuit.....  | 120 |
| Figure 6. 4 Example of Power Consumption Information Leakage [90].....                        | 122 |
| Figure 7. 1 Memory model for Tiny OS on MICA mote .....                                       | 141 |
| Figure 7. 2 Proposed Algorithm of Elastic Window for Scalar Multiplication on WSN Nodes ..... | 144 |
| Figure 8. 1 Man-in-Middle Attack in WSN .....   | 147 |
| Figure 8. 2 Distribution of Elliptic Group $E_{23}(1, 1)$ . .....                             | 150 |
| Figure 8. 3 A new protocol protecting the man-in-the-middle attack. ....                      | 150 |

Figure 8. 4 Block diagram for hidden generator point principle. .... 152

Figure 8. 5 A protocol for ECC with hidden generator point ..... 152

Figure 8. 6 Block Diagram of MAS Framework for ECC..... 155

Figure 11. 1 Cloud Computing Architecture for WSN..... 165

## List of Algorithms

---

|  |     |
|--|-----|
| Algorithm 3. 1 Elliptic Curve key pair generation .....                          | 46  |
| Algorithm 3. 2 Elliptical Curve Diffie-Hellman Protocol.....                     | 48  |
| Algorithm 3. 3 Basic ElGamal Encryption Algorithm.....                           | 53  |
| Algorithm 3. 4 Basic ElGamal Decryption Algorithm .....                          | 53  |
| Algorithm 3. 5 Point Addition .....  | 57  |
| Algorithm 3. 6 Point Doubling.....   | 58  |
| Algorithm 3. 7 Reduction Modulo .....  | 60  |
| Algorithm 3. 8 Barrett Reduction Algorithm.....                                  | 62  |
| Algorithm 3. 9 Montgomery Reduction Simple Case .....                            | 62  |
| Algorithm 3. 10 Modified Montgomery Reduction .....                              | 63  |
| Algorithm 3. 11 <i>Computing <math>x^{-1}(\text{mod } 2^w)</math></i> .....      | 64  |
| Algorithm 3. 12 Montgomery Multiplication.....                                   | 65  |
| Algorithm 3. 13 Reduction Modulo .....   | 67  |
| Algorithm 4. 1 Point Addition and Point Doubling in Affine Coordinate.....       | 78  |
| Algorithm 4. 2 Point Addition and Point Doubling in Projective Coordinate .....  | 79  |
| Algorithm 4. 3 Point Addition and Point Doubling in Jacobian Coordinate.....     | 81  |
| Algorithm 4. 4 Conversion of Affine coordinates to other coordinates .....       | 83  |
| Algorithm 4. 5 Conversion of projective coordinate to other coordinates[4] ..... | 83  |
| Algorithm 4. 6 Conversion of Jacobian coordinates to other coordinates[4].....   | 84  |
| Algorithm 5. 1 Left to Right Binary Method.....                                  | 94  |
| Algorithm 5. 2 Right to Left Binary Method.....                                  | 94  |
| Algorithm 5. 3 Computing the NAF of a positive integer.....                      | 97  |
| Algorithm 5. 4 Binary NAF method for point multiplication .....                  | 98  |
| Algorithm 5. 5 Conversion from Binary to NAF .....                               | 98  |
| Algorithm 5. 6 Left to Right Processing from Binary to MOF .....                 | 100 |
| Algorithm 6. 1 SPA Resistant Double and Add Always Method .....                  | 125 |
| Algorithm 6. 2 Montegomery Ladder to avoid SPA [26].....                         | 126 |
| Algorithm 6. 3 Window OCS method for scalar multiplication .....                 | 128 |
| Algorithm 7. 1 Multiplication by m-ary method.....                               | 133 |
| Algorithm 7. 2 Sliding Window method.....  | 134 |
| Algorithm 8. 1 Proposed Protocol to avoid man in the middle attack .....         | 153 |
| Algorithm 9. 1 Legendre Symbol.....  | 158 |
| Algorithm 9. 2 Square root modulo p .....  | 159 |

## List of Tables

---

|   |     |
|---|-----|
| Table 2. 1 Family of Berkeley Nodes [9] .....   | 18  |
| Table 2. 2 Specifications of First Generation Nodes [1].....                              | 19  |
| Table 2. 3 Specifications of Imote2 [10] .....  | 20  |
| Table 3. 1 Nomenclatures used in ECC .....  | 42  |
| Table 3. 2 NIST Guidelines for Key Sizes .....  | 49  |
| Table 4. 1 Comparison of binary and prime field on Motorola processor .....               | 75  |
| Table 4. 2 Cost of Conversion To and From Various Coordinate Systems [4] .....            | 84  |
| Table 4. 3 Cost of Operations In Mixed Coordinates [24] .....                             | 86  |
| Table 4. 4 Representation of Point in Various Coordinate Systems[24].....                 | 87  |
| Table 4. 5 Comparison of Projective and Affine Coordinate for ECDH and ECDSA for WSN..... | 88  |
| Table 5. 1 Comparison of OCS with other methods on MATLAB.....                            | 116 |

## Abbreviations

---

The following abbreviations of standard phrases are used throughout the thesis:

|                |  |
|----------------|--|
| <b>AES</b>     | Advanced Encryption Standard                                 |
| <b>AES 128</b> | AES with a 128-bit key                                       |
| <b>AES 192</b> | AES with a 192-bit key                                       |
| <b>AES 256</b> | AES with a 256-bit key                                       |
| <b>CM</b>      | Complex Multiplication                                       |
| <b>CMOS</b>    | Complementary Metal Oxide Semiconductor                      |
| <b>DES</b>     | Data Encryption Standard                                     |
| <b>DLP</b>     | Discrete Logarithm Problem                                   |
| <b>DoS</b>     | Denial-of-Service  |
| <b>DSA</b>     | Digital Signature Algorithm                                  |
| <b>ECC</b>     | Elliptical Curve Cryptography                                |
| <b>ECDH</b>    | Elliptical Curve Diffie-Hellman                              |
| <b>ECDLP</b>   | Elliptical Curve Discrete Logarithmic Problem                |
| <b>ECDSA</b>   | Elliptical Curve Digital Signature Algorithm                 |
| <b>GCD</b>     | Greatest Common Divisor                                      |
| <b>IEEE</b>    | Institute of Electrical and Electronics Engineers            |
| <b>ISM</b>     | Industrial, Scientific and Medical band                      |
| <b>JSF</b>     | Joint Sparse Form  |
| <b>μTESLA</b>  | Micro timed Efficient Stream Loss Tolerant Authentication    |
| <b>MAC</b>     | Media Access Control   |
| <b>MEMS</b>    | Micro Electro Mechanical Systems                             |
| <b>MIRACL</b>  | Multiprecision Integer and Rational Arithmetic C/C++ Library |
| <b>NAF</b>     | Non Adjacent Form  |
| <b>NIST</b>    | National Institute of Standards and Technology               |

|             |                                       |
|-------------|---------------------------------------|
| <b>ONB</b>  | Optimal Normal Basis                  |
| <b>PKC</b>  | Public Key Cryptography               |
| <b>RAM</b>  | Random Access Memory                  |
| <b>ROM</b>  | Read Only Memory                      |
| <b>RSA</b>  | Rivest Shamir Adleman algorithm       |
| <b>SD</b>   | Signed Digits                         |
| <b>SNEP</b> | Sensor Network Encryption Protocol    |
| <b>SPA</b>  | Simple Power Analysis                 |
| <b>SPIN</b> | Security Protocol for Sensor Networks |
| <b>VLSI</b> | Very Large Scale Integration          |
| <b>WSN</b>  | Wireless Sensor Networks              |



## Acknowledgements

---

My sincere thanks to Hon'ble Prof. Dharmendra Sharma, Dean of Faculty of Information Sciences and Engineering, University of Canberra who offered me opportunity to do research in Australia's one of the best universities.

Special gratitude is to PhD Supervisor and Chair Prof. Dr. Xu Huang ,who's faith in my work and his constant support at all academic and research levels were priceless factors that helped me to achieve more than I thought possible.

Last but not least, I would like to thanks my parents, wife Rakhee, daughters Darshana and Namita for their endless support and understanding through last five years of my research. Their love and confidence in me were the constant source of inspiration to offer the best of myself to this research.

Submitted Please.

**Mr. Pritam Gajkumar Shah**



## Chapter 1 Introduction

---

### 1.1 Background and Motivation

Recent advances in Micro-Electro-Mechanical Systems (MEMS) technology, wireless communications, and digital electronics have enabled the development of low-cost, low-power, multifunctional micro sensor nodes that are small in size and communicate in short distances [1-4].

A WSN is a network of such sensors that can (a) sense specified parameters relating to their environment, (b) process them either locally or in a distributed manner, and (c) communicate the processed information to ‘base station’ which in turns communicates with one or more central processing centers (CPCs). The CPCs are expected to analyze the information and respond suitably.

As a result one anticipates that WSN could be effectively deployed to provide early-warning systems, or post-event notification in various sectors, including environment, forestry (read forest fires), agriculture (read precision farming), national security (intrusion detection along international borders, or tracking hostile objects), public health in work places (mines, nuclear power plants, refineries), and disaster mitigation [2].

As WSN handle sensitive data and operate in a hostile unattended environment, it is crucial that security concerns be addressed from the beginning of the system design. However, due to inherent resource and computing constraints, security in WSN poses different challenges than traditional network security [5]. It is difficult to directly employ existing security

approaches to the area of WSN. Therefore, to develop useful security mechanisms while borrowing the ideas from the current security techniques, it is necessary to know and understand major obstacles for WSN such as limited processing power, storage, bandwidth and energy. They are discussed thoroughly in the chapter 2 of this thesis. By design WSN nodes are inexpensive, low power devices. As a result, current state-of-the-art protocols and algorithms, which are doing a satisfactory job of securing communication, will be too heavy weight for use in WSN. These algorithms are having very high communication overheads and are not designed to run on computationally constrained devices. So there is a need for new energy efficient cryptographic algorithms and protocols. While implementing cryptographic protocols on WSN, two approaches are possible one is *symmetric key* and other is *public key*.

There is common perception about public key cryptography (PKC) [6, 7] that it is complex, slow and power hungry, and as such not at all suitable for use in ultra-low power environments like WSN [8]. It is therefore common practice to imitate the asymmetry of traditional public key based cryptographic services through a set of protocols using symmetric key based message authentication codes (MACs) [9]. Although the low computational complexity of MACs is advantageous, the protocol layer requires time synchronization between devices on the network and a significant amount of overhead for communication and temporary storage. These requirements make 8 bit CPU of WSN node prone to vulnerabilities and practically eliminates all the advantages of using symmetric key techniques in the first place. There arise need for investigation which will make PKC feasible in WSN environments, provided we use the right selection of algorithms and associated parameters, careful optimization, and low-power design techniques. Two of the major techniques are used to implement public-key cryptosystems are RSA [6] and Elliptic Curve Cryptography (ECC) [7]. Traditionally, these have been thought to be far too heavy weight for use in WSN. But the recent research efforts made by [10-13] demonstrated that ECC is feasible for small networks with careful design of algorithms and pointed out the compact key size ECC is promising technology for all light weight applications including WSN.

Gura et.al [13] implemented ECC on an 8 bit microcontroller by using elliptic curves over prime integer field  $GF(p)$  and showed that ECC is practically feasible on 8-bit microcontroller . Elliptic curves over prime fields were chosen since binary polynomial field arithmetic was insufficiently supported by cotemporary microprocessors and would have lead to lower performance. The point scalar multiplication process was decomposed into sequence of point additions and point doublings and proposed hybrid method for multiplication. The curves referred were standardised by NIST [14] . The mixed coordinate system was used to achieve better results. The Non Adjacent Forms (NAF) method [15] was used for recoding the scalar  $k$  in point multiplication.

Shantz et. al [16] presented an efficient technique to calculate a modular division .The idea is to compute  $y/x$  in one operation instead of computing  $1/x$  first and then multiplying it with  $y$  . This scheme has reduced one multiplication in modular division operation. The new algorithm can be applied in both prime as well as binary field.

Woodbury et. al [17] introduced another ECC system over *optimal extension field* (OEF)  $GF(p^m)$  where  $p$  is chosen of the form  $2^n \pm c$  . The author implemented 134 ECC in less than 2 seconds.

Cohen et. al [18] analysed the impact of a coordinate system and proposed new modified Jacobean coordinates which achieves the fastest doubling operation. Moreover they introduced a mixed coordinate system, which divides exponentiation into sub operations and chose the best coordinate representation for each sub operation.

Malan et al. [12] implemented ECC over binary extension field curves as it allows space and time efficient algorithms. Implementation was carried out on MICA2 platform with Tiny OS

and *nes C* language. They used polynomial basis and multiplication of point was obtained by binary method [15]. Addition was achieved with Algorithm 7 and multiplication of elements in  $GF(2^p)$  was implemented as Algorithm 4 in Lopez and Dahab [19] while inversion was implemented as Algorithm 8 in Hankerson et. al[20]. Mallan's [12] EccM2.0 obeys NIST recommended curves [14] over *binary extension field*.

The major points noted from the above *literature survey* are listed below and forms the foundation of this thesis.

1. The applications involved in the above approaches are varying in terms of hardware and software platforms. ECC is highly platform dependent .Algorithm written for one platform may not work well on other due to various computational and memory requirements.
2. Most of the researchers have used borrowed algorithms from the mathematical literature of ECC. The choice of proper algorithm is the key issue. The approach followed by this research is to develop algorithm exclusively for WSN by taking in to account its resource limitations including hardware as well as software.
3. Due to VLSI advancement there is an unprecedented growth in the hardware technology. For example, the previous generations MICA mote were having 512 bytes RAM that's has gone up to 256 Kbytes in Imote2, the processor speed has scaled up from 4 MHz to 416 MHz [21] .These changes will definitely going to affect the performance of ECC on the new platform.

Due to above facts optimization of ECC for light weight applications like WSN is an *open research issue*.

## **1.2 Research Questions**

Before implementing ECC on WSN, several selections have to be made concerning the finite field, elliptical curves and cryptographic protocol:

1. A finite field, a representation for the field elements, and algorithms performing the field arithmetic.
2. An elliptical curve, a representation for the elliptical curve points, and algorithms for performing elliptic curve arithmetic.
3. A protocol, and algorithms for performing protocol arithmetic.

There are many factors that can influence the choices to be made. All of these factors must be considered simultaneously in order to arrive at the best solutions for WSN. Relevant factors include security considerations, application platform (software and hardware), constraints of the particular computing environments e.g. processing speed, code size (ROM), memory size (RAM), gate count, and constraints of the particular communication environment e.g. bandwidth, response time.

Not surprisingly, it is difficult, if not impossible, to decide on a single “best” set of choices. For example, the optimal choices for ‘base station’ of WSN can be quite different from the optimal choices for WSN node or smart card application. This research will provide a comprehensive account of the various algorithms and security considerations for ECC, so that informed decisions of the most suitable options can be made for WSN.

The further objective of this thesis is to investigate “*acceleration of ECC*” on WSN platform by answering the following research question:

All standard ECC protocols like Elliptical Curve Diffie Hellman (ECDH) [22], Elliptical Curve Digital Signature Generation (ECDSA) [23] used for *pair wise key establishment* and *authentication* respectively requires scalar multiplication operation to calculate public key. Scalar multiplication occupies 80% of the protocol execution

time and depends on several factors. Which are the various factors influencing the scalar multiplication time and how to accelerate scalar multiplication process by keeping memory, bandwidth requirements of WSN under control?

This research question will be further investigated by looking at the following relevant sub questions.

1. Which is the optimal method to recode integer while doing scalar multiplication on the WSN node? The question will address all relevant hardware and software issues involved in this process.
2. Which is the optimal coordinate system to represent the points on the elliptical curve on WSN node so that inversion operation can be avoided to accelerate the scalar multiplication process?
3. How to avoid Special Power Analysis (SPA) attacks by making suitable changes in the binary method of scalar multiplication algorithm on WSN node?
4. How to prevent node failures while doing scalar multiplication by selection of optimal window size depending on the available memory (RAM)?
5. What are the precautions to be taken to quash *man-in-the-middle-attack* while implementing ECDH protocol for pair wise key establishment among WSN nodes?
6. Is there any possibility of using only one coordinate ( $x$  or  $y$ ) to represent point on elliptical curve so that WSN node can save bandwidth and avoid communication overheads?

### **1.3 Research Methodology**

To validate merits of proposed algorithms this thesis has extensively utilized **M**ultiprecision **I**nteger and **R**ational **A**rithmetic **C**/C++ **L**ibrary (MIRACL) [24] and MATLAB software



[25]. Results obtained on the MIRACL are included in the thesis appendix and at places wherever necessary.

MIRACL is a Big Number Library which implements all of the primitives necessary to design Big Number Cryptography into your real-world application. It is primarily a tool for cryptographic system implementers. Elliptic Curve Cryptography (ECC), Diffie-Hellman (D-H) Key exchange, DSA digital signature, they are all just a few procedure calls away. The MIRACL offers full support for Elliptic Curve Cryptography over  $GF(p)$  and  $GF(2^m)$ . Less well-known techniques can also be implemented as MIRACL allows you to work directly and efficiently with the big numbers that are the building blocks of number-theoretic cryptography. Although implemented as a C library, a well-thought out C++ wrapper is provided, which greatly simplifies program development. MIRACL is compact, fast and efficient and it's now easier than ever to get the same near-optimal performance from any processor. Although essentially a portable library, inline assembly and special techniques can be invoked for blistering speed. MIRACL has also been successfully used in both embedded and DSP environments where space is at a premium. A new special purpose macro assembler feature facilitates the achievement of best possible performance from embedded processor [24].

## **1.4 Research Contribution**

The objective of this research was to investigate major challenges involved in implementing Elliptical Curve Cryptography (ECC) for Wireless Sensor Networks (WSN) and to develop sustainable security model based on it. This objective is achieved by using five way approaches. The main contributions of this thesis are as per below.

Firstly, this research has proposed algorithm based on one's complement subtraction (OCS) to recode integer in scalar multiplication. This algorithm will offer less Hamming weight which reduces number of point operations and remarkably improve computational efficiency of scalar multiplication. Less point operations guarantee faster key calculation time. As compared with existing best available method Non Adjacent Form (NAF), this algorithm requires only bitwise subtractions to recode integer and is hardware friendly for WSN platform. The effectiveness of this proposed recoding method can be enhanced in combination with sliding window method with flexible window size. This work resulted in a publication and an innovative patent. Details can be obtained in Appendix A.

Secondly, possibilities of *simple power analysis* (SPA) attack on binary method is presented and modified binary method resistant to SPA attack has been proposed exclusively for WSN platform which do not requires any hardware or architectural changes and can be implemented on any wireless sensor node at software level by utilization of *exponent splitting* technique. This work has been underpinned by an innovative patent and is under consideration by Australian patent office. (Please refer Appendix A)

Thirdly, *Elastic window* method has been proposed to accelerate the scalar multiplication process for WSN nodes. Sliding window method consists of two stages namely pre computation stage and an evaluation stage. Points for use in the evaluation stage are computed in the pre computation stage. The scalar multiplication is carried out in the evaluation stage with the addition of pre computed points. The number of pre computations depends on the window size of sliding window method. More is the window size, more are the pre computations and more is the memory required for the storage. This is the well-known draw-back of the sliding window method when implemented on WSN nodes. This research recommended *Elastic Window (EW)* method with adjustable window size for scalar multiplication on wireless sensor nodes. The EW will prevent WSN node failure due to stack overflow. This work has been underpinned by a publications and an innovative patent.

Fourthly, a robust protocol based on hidden generator point is proposed to avoid the *man-in-the-middle-attack* in the WSN architecture. As the base point of elliptical curve is in public domain any adversary node can try to fetch information from other nodes if the authentication of the nodes is not done by trusted authority. To avoid this problem which is well known drawback of ECDH protocol, this research proposed a protocol based on hidden base point of elliptical curve. This protocol is provided only for academic interest and may be useful in the future as technology for WSN advances. This work has resulted in a publications and an innovative patent.

Finally, a scheme based on uni-coordinate is suggested for WSN which will save bandwidth and transmission power of WSN node considerably. This scheme make use of concept of solving quadratic equation in the context of elliptical curves, where it is used to obtain the y-coordinate of a point if x-coordinate is given. In this new proposed scheme node will transmit only x -coordinate and the receiving node will calculate the y- ordinate by using algorithms like Legendre symbol to obtain the public key.

## **1.5 Thesis Outline**

The remainder of the thesis is organized as follow. Chapter 2 gives literature review of WSN and its node architecture, node power consumptions, WSN topology, WSN protocol stack, routing protocols for information exchange, possible security threats classified layer wise, a small survey of cryptographic algorithms and scope for future improvements.

Chapter 3 provides ECC modelling, standard protocols based on ECC, finite field arithmetic and types of bases, algorithms for modular inversions, point addition, point doublings.

Chapter 4 analyses the various coordinate systems available for an ECC on WSN in terms of efficiency, code size, memory footprint and also includes recommendation for the selection of coordinate system on WSN platform. The results obtained on MIRACL crypto library are included at the end of chapter.

Chapter 5 provides description and analysis of *proposed algorithm of Ones' Complement Subtraction (OCS)* for recoding of integer in scalar multiplication process for WSN platform and its comparison with existing methods like NAF, MOF. The results obtained on MATLAB are included at the end of the chapter.

Chapter 6 gives overview of existing side channel attacks for e.g. simple power analysis (SPA), differential power analysis (DPA), existing counter measures for side channel attacks like double and add always etc and proposes SPA resistant algorithm for WSN.

Chapter 7 proposed a new *Elastic window method* for scalar multiplication process on WSN which will avoid node failures due to stack overflow problem. It also gives brief introduction of the existing scalar multiplication methods and their limitations for WSN.

Chapter 8 proposed a new protocol based on concept of hidden base generator to avoid man in the middle attack and has given framework of multi agent system (MAS) based on it for WSN security. This chapter contains analysis of its efficiency and security proof of protocol when implemented on WSN platform.

Chapter 9 provides mathematical modelling of a proposed new scheme of uni-coordinate for WSN in which the node will transmit single coordinate of the public key and the other coordinate is calculated by the receiving WSN node. This chapter also provides the various advantages of this scheme for the WSN.

Chapter 10 conclude thesis by summarizing main achievements of the research and contribution to knowledge.

Chapter 11 will outline areas of possible future research.

Appendices are provided at the end of thesis containing examples of standard NIST curves, results obtained on the MIRACL crypto library, list of author's patents and publications. The Bibliography is provided at the end in the IEEE sensor network format.

## **1.6 Summary**

Node authentication and key management are two major aspects of WSN security as nodes are scattered in hostile and unattended environments. In traditional networks such as the Internet, Public Key Cryptography (PKC) has been the enabling technology underlying many security services and protocols e.g., SSL[26] and IPsec [27]. However, due to the resource constraints nature of WSN nodes such as limited battery power, bandwidth and computational capabilities, PKC has not been widely adopted.

There has been intensive research aimed at developing techniques that can bypass PKC operations in WSN such as random key pre-distribution for pair wise key establishment [28-32] and broadcast authentication [33-35]. However, these alternative approaches do not offer the same degree of security or functionality as PKC.

For instance, none of the random key pre-distribution schemes can guarantee key establishment between any two nodes and tolerate arbitrary node compromises at the same time. As another example, the existing broadcast authentication schemes, which are all based on TESLA [55], requires loose time synchronization, which itself is a challenging task to achieve in wireless sensor networks.

In contrast, PKC can address all these problems easily. Pair wise key establishment can always be achieved using, the Diffie-Hellman (DH) key exchange protocol [22], without suffering from the node compromise problem. Similarly, broadcast authentication can be provided with, the ECDSA digital signature scheme [23], without requiring time synchronization. Thus, it is desirable to explore the application of PKC on resource constrained sensor platforms.

There have been a few recent attempts [35-37] to use PKC in WSN, which demonstrate that it is feasible to perform limited PKC operations on the current sensor platforms such as Telos nodes.

ECC has been the top choice among various PKC options due to its fast computation, small key size, and compact signatures. As said earlier, to provide equivalent security to 1024-bit RSA, an ECC scheme only needs 160 bits on various parameters, such as 160-bit finite field operations and 160-bit key size [38].

Despite the recent progress on ECC implementations on WSN platforms, all the previous attempts [35, 37, 39] have limitations. As quoted by [40], all these attempts were developed as independent packages and / or applications without seriously considering the resource demands of WSN applications. As a result, developers may find it difficult, and sometimes impossible, to integrate an ECC implementation with WSN applications, though the ECC implementation may be okay on its own. For example, an ECC implementation may require so much RAM that it is impossible to fit both the WSN application and the ECC implementation on the same WSN node.

Moreover, various optimization techniques are available to speed up the ECC operations. Such optimizations, however, typically will increase the ROM and RAM consumptions, though they may reduce the execution time and energy consumption. It is not clear what optimizations should be used and how they should be combined to achieve the best trade-off

among security protection, computation overheads, and storage requirements. Additional research is necessary to clarify these issues and facilitate the adoption of ECC-based PKC in WSN[40].

Based on this background, the research question in this thesis was developed as:

All standard ECC protocols like Elliptical Curve Diffie Hellman (ECDH) [22], Elliptical Curve Digital Signature Generation (ECDSA) [23] used for WSN node pair wise key establishment and authentication respectively requires scalar multiplication operation to calculate public key. Scalar multiplication occupies 80% of the protocol execution time and depends on several factors. Which are the various factors influencing the scalar multiplication time and how to accelerate scalar multiplication process by keeping memory, bandwidth requirements of WSN under control?

To tackle this research question, various innovative algorithms which will improve scalar multiplication process, on WSN nodes will be discussed in the thesis Chapter 4 to Chapter 9.

## Chapter 2 Wireless Sensor Networks and Security: A Review

---

### 2.1 Introduction:

Wireless Sensor Networks (WSN) is a special case of mobile adhoc networks (MANET) and do not have any formal infrastructure. Thousands of sensor nodes are scattered in the field with average distance of about 5 meter. The node density can go up to 20 nodes/m<sup>3</sup>. Sensor nodes can be thrown in mass or can be deployed manually in the field. Once they are deployed they form the network and communicate with *single hop* or *multi hop* with radio frequencies in the range of ISM band (916 MHz) [41]. The nodes collect the information and pass it to the base station or sink. Base station in turn communicates with user via satellite or internet. Figure 2.1 shows a typical WSN architecture .

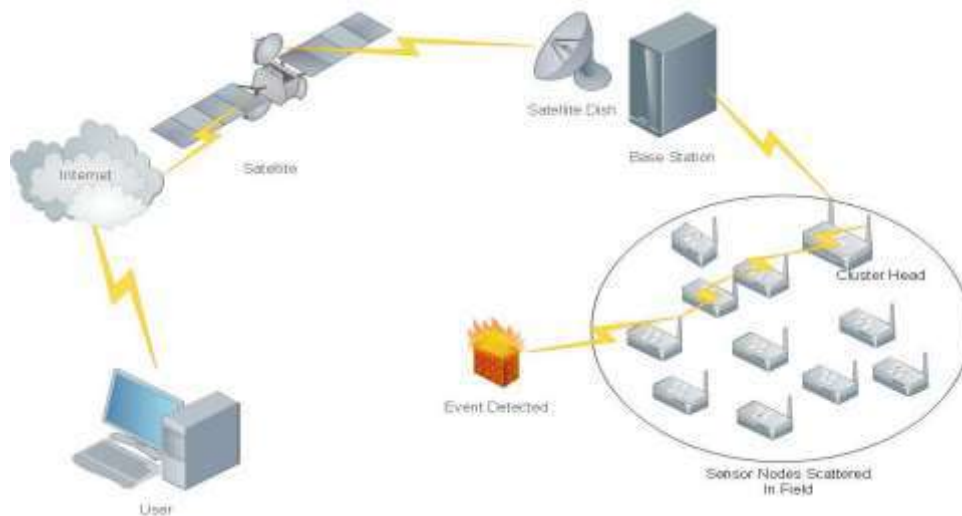


Figure 2. 1 A Typical WSN Architecture



WSN can be also deployed in many other applications such as environmental monitoring [42], biomedical research [43], human imaging and tracking [44, 45], and military applications [46]. As per vision of [41], WSN are slowly becoming an integral part of our lives.

Before discussing single node architecture it is wise to understand the major differences between WSN and MANET, which are as follows-

1. The node density in WSN is very high as compared to MANET.
2. Sensor nodes are deployed in thousand numbers all over the field but the MANET may not be necessarily like this.
3. There is always possibility of node dying or failures due to battery problem or other reasons for WSN but for MANET will not be normally the case.
4. The topology of a sensor network is always changing due to dying of few nodes or addition of the new ones but for MANET for fixed area topology may not necessary to be frequently changed.
5. Sensor nodes usually transmit information with broadcast mechanism whereas ad hoc networks transmit information by relay based on point- to-point communications.
6. As the node density is very high in WSN, node does not have identification numbers or physical addresses but for MANET there is always has ID with the address to be identified.
7. In WSN there is always "Cluster Head" to collect the information from the cluster and then passing to the back-end database but for MANET each node can talk to others in particular ad hoc fashion.

## 2.2 WSN Architecture

### 2.2.1 WSN Node

Every sensor node is made up of four basic building blocks namely sensors, microcontroller, transmitter and receiver and battery as shown in figure 2.2. They may also have some optional components like location finding system, solar panel or piezo-electric mechanism for recharging of batteries and mobilizer.

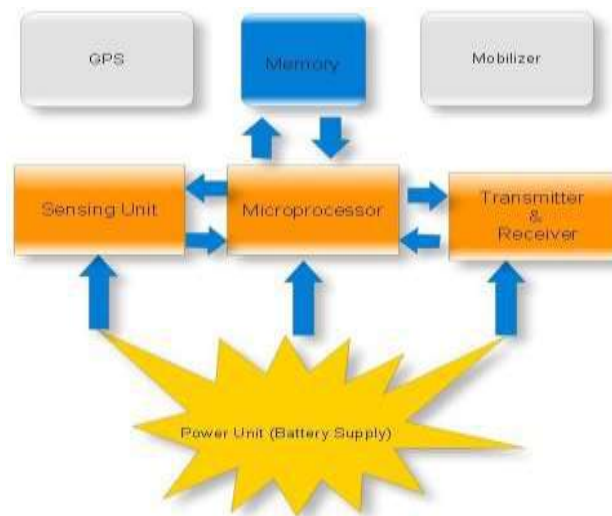


Figure 2. 2 Block Diagram of WSN Node

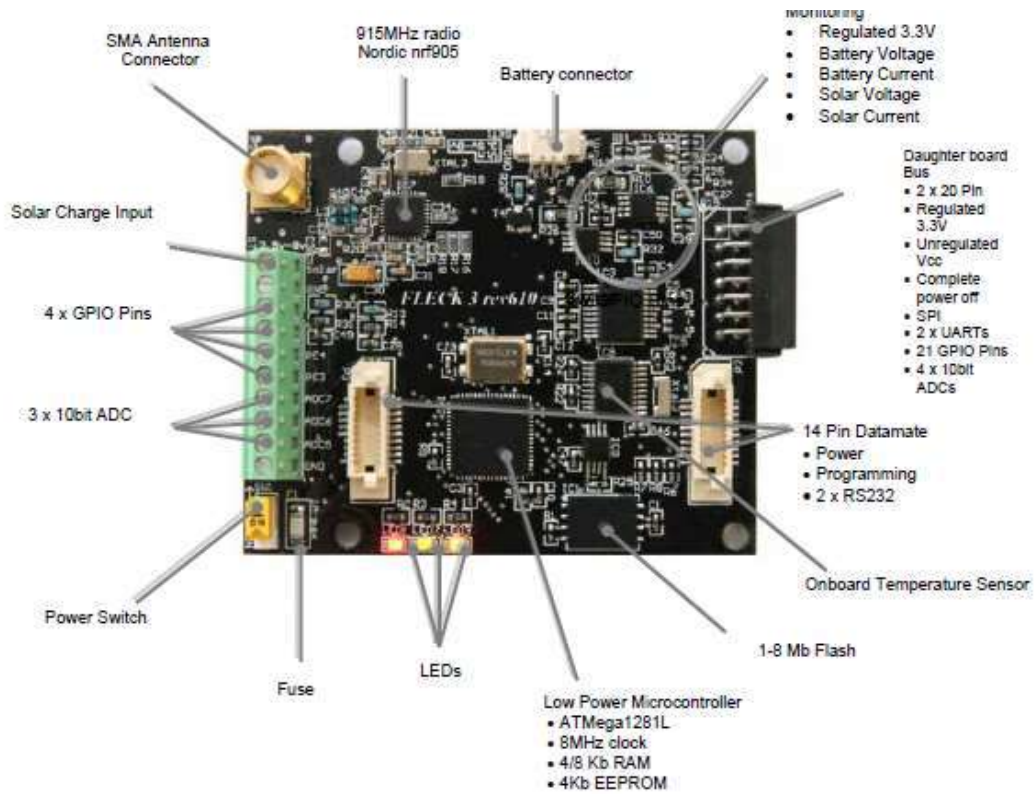


Figure 2. 3 Fleck3B WSN Node by CSIRO [7]

The figure 2.3 shows Fleck node manufactured by CSIRO [47] having long range radio and powerful interfacing capacity. The figure 2.4 shows wireless nodes manufactured by University of Berkeley in tabular form with their specifications. Figure 2.5 shows MICAz node by Crossbow Inc [21] and Telos motes by Berkeley [48] respectively.

Table 2. 1 Family of Berkeley Nodes [9]

| Mote Type<br>Year                | WeC<br>1998  | René<br>1999 | René2<br>2000 | Dot<br>2000 | Mica<br>2001 | Mica2Dot<br>2002 | Mica 2<br>2002 | Telos<br>2004 |
|----------------------------------|--|--------------|---------------|-------------|--------------|------------------|----------------|---------------|
| Microcontroller                  |  |              |               |             |              |                  |                |               |
| Type                             | AT90LS8535   |              | ATmega163     |             | ATmega128    |                  | TI MSP430      |               |
| Program memory (KB)              | 8  |              | 16            |             | 128          |                  | 48             |               |
| RAM (KB)                         | 0.5  |              | 1             |             | 4            |                  | 10             |               |
| Active Power (mW)                | 15   |              | 15            |             | 8            |                  | 33             |               |
| Sleep Power ( $\mu$ W)           | 45   |              | 45            |             | 75           |                  | 75             |               |
| Wakeup Time ( $\mu$ s)           | 1000   |              | 36            |             | 180          |                  | 180            |               |
| Nonvolatile storage              |  |              |               |             |              |                  |                |               |
| Chip                             | 24LC256  |              |               |             | AT45DB041B   |                  | ST M25P80      |               |
| Connection type                  | I <sup>2</sup> C   |              |               |             | SPI          |                  | SPI            |               |
| Size (KB)                        | 32   |              |               |             | 512          |                  | 1024           |               |
| Communication                    |  |              |               |             |              |                  |                |               |
| Radio                            | TR1000   |              |               |             | TR1000       |                  | CC1000         |               |
| Data rate (kbps)                 | 10   |              |               |             | 40           |                  | 38.4           |               |
| Modulation type                  | OOK  |              |               |             | ASK          |                  | FSK            |               |
| Receive Power (mW)               | 9  |              |               |             | 12           |                  | 29             |               |
| Transmit Power at 0dBm (mW)      | 36   |              |               |             | 36           |                  | 42             |               |
| Power Consumption                |  |              |               |             |              |                  |                |               |
| Minimum Operation (V)            | 2.7  |              | 2.7           |             | 2.7          |                  | 1.8            |               |
| Total Active Power (mW)          | 24   |              | 27            |             | 44           |                  | 89             |               |
| Programming and Sensor Interface |  |              |               |             |              |                  |                |               |
| Expansion                        | none   | 51-pin       | 51-pin        | none        | 51-pin       | 19-pin           | 51-pin         | 16-pin        |
| Communication                    | IEEE 1284 (programming) and RS232 (requires additional hardware) |              |               |             |              |                  |                | USB           |
| Integrated Sensors               | no   | no           | no            | yes         | no           | no               | no             | yes           |

As most of the sensors are having analog output, hence analog to digital converters (ADC) are used for digitizing the information and given to the microcontroller for processing. The microcontroller on the nodes consists of flash, data registers and scratch pad memory for storing the monitor program which performs functions such as processing of the information given by sensors, collaborating with neighboring nodes and base station etc. Due to advancement in VLSI technology all these modules are fabricated on a structure which is having a size of cubic centimeter. Wireless sensor node consumes very low power. The current drawn is always in the range of  $\mu A$  to  $mA$  depending on the state of sensor. Nodes are powered from typical Lithium batteries.



Figure 2. 4 Images of MICA Node (on Left) and Telos Node (on Right)

Table 2. 2 Specifications of First Generation Nodes [1]

| <i>Sr. No</i> | <i>Parameter</i>            | <i>Value</i>  |
|---------------|-----------------------------|---|
| 1             | CPU                         | 8 Bit,4 MHz,8 Kbytes Instruction Flash,<br>512 Bytes RAM, 512 Bytes ROM |
| 2             | Communication               | 916 MHz   |
| 3             | Bandwidth                   | 10 Kbps   |
| 4             | Operating System            | Tiny OS   |
| 5             | Operating System Code Space | 3500 Bytes  |
| 6             | Available Code Space        | 4500 Bytes  |

**Table 2. 3 Specifications of Imote2 [10]**

| <i>Sr. No</i> | <i>Parameter</i>      | <i>Value</i>   |
|---------------|-----------------------|--|
| 1             | CPU                   | Intel PXA271, 13MHz to 416MHz ,<br>32 Mbytes Instruction Flash<br>256 Kbytes SRAM ,32 Mbytes SDRAM   |
| 2             | Communication         | 2400.0 – 2483.5 MHz  |
| 3             | Bandwidth             | 250 kb/s   |
| 4             | Operating System      | Tiny OS, Linux or SOS  |
| 5             | Range (line of sight) | ~30 m With integrated antenna  |
| 6             | Power Source          | Battery Board 3x AAA<br>USB Voltage 5.0 V<br>Battery Voltage 3.2 – 4.5 V<br>Li-Lon Battery Charger   |
| 7             | Power Consumption     | Current Draw In Deep Sleep Mode 390 $\mu$ A<br>Current Draw In Active Mode 31 mA 13MHz, radio off<br>Current Draw In Active Mode 44 mA 13MHz, radio Tx/Rx<br>Current Draw In Active Mode 66 mA 104MHz, radio Tx/Rx |

Specifications of Imote 2 are shown in Table 2.3 is the seventh generation wireless node (Please refer figure 2.4 for generations of nodes) and is having up to date resources. From these specifications following conclusions may be drawn,

- IMote2 consists of Intel microprocessor which works on the frequency range of 13 MHz to 416 MHz which is at considered to be at low end from point of view of public key cryptography implementations. The calculating ECC key is computationally intense process.
- The memory available is only 256 Kbytes (SRAM). It indicates that there is small space for storing 'stack' and pre-computed values during calculation of public key.
- The flash is of the size of 32 MB which stores operating system for the wireless sensor node as well as programme code for implementation of ECC.
- Imote 2 consumes maximum current during transmission about 44 mA at 13 MHz with data rate of 250 kb/s which underlines the need of small key size during the execution of cryptographic protocol.
- Apart from these the battery which is available on the board is of 3.3.V with 2A-Hr capacity. It implies that every algorithm used in WSN security must be power wise.

Due to the above boundaries the current state of art protocols and algorithms are expensive for WSN due to their high communication overheads such as key size, key calculation time. These protocols were not designed to run on computationally constrained devices and require high computational power, memory and energy sources to run satisfactorily. As WSN can not

afford this luxury, there is a need for new cryptographic algorithms and protocols for WSN which will be light weight.

### 2.2.2 Node Power Consumption

Battery life plays important role in the function of the sensor node. When the sensor is in active mode and transmitting information, it draws current of the order of 41mA (Table 2). Therefore sensor node life time shows strong dependence on battery life time. In multi hop network every sensor node acts as originator as well as router of the information. So failure of node can cause rerouting of the packets and major changes in the network topology. This is the primary task of this research to provide power aware protocols and algorithms for WSN security. In other networks power consumption is important factor but not primary consideration but in case of WSN power consumption directly influences the life time of network. Power consumption in WSN is attributed to three main domains namely *sensing*, *communication*, and *data processing*. As this research is related to WSN security it will concentrate on only communication and data processing domain. Calculating public key shows strong dependence on memory and computational speed of microcontroller.

**a) Power required for Communication:** Wireless sensor Node spends maximum energy on transmitting and receiving the information. As seen from the Table 2.3 , transmitting and receiving current requirements are nearly same for short range communication. Mixers, frequency synthesizers, voltage control oscillators (VCO), phase locked loops (PLL) and power amplifiers; all consume valuable power in the transmitter and receiver circuit. The power required for turning sensor node from sleeping to active mode is

non-negligible. The radio power consumption can be given by formula[49],

$$P_c = N_T[P_T(T_{ON} + T_{ST}) + P_{OUT}(T_{ON})] + N_R[P_R(R_{ON} + R_{ST})] \dots\dots\dots(2.1)$$

Where,



$P_T$  is the power consumed by the transmitter/receiver,

$P_{OUT}$  is the output power of the transmitter,

$T_{ON}$  is the transmitter/receiver on time,

$R_{ST}$  is the transmitter/receiver startup time and

$R_{ON}$  is the number of times transmitter/receiver is switched on per unit time,

$T_{ON}$  can further be rewritten as  $L / R$ , where  $L$  is the packet size and  $R$  is the data rate.

**b) Power required for data processing:** Power consumed for data processing is much less as compared energy spent on communication. Assuming Rayleigh fading and fourth order power distance loss, the energy cost of transmitting 1Kb a distance of 100 m is approximately the same as that for executing 3 million instructions by a 100 million instructions per second (MIPS)/W processor. Due to cost and size limitations all the motes are built with CMOS (Complementary Metal Oxide Semiconductor) technology. CMOS transistors are power consuming and draw plenty of current when transistor goes to active state from cut off state. The power consumption in data processing can be given by formula[50],

$$P_p = CV_{dd}^2 f + V_{dd} I_o e^{V_{dd}/nV_T} \dots\dots\dots(2.2)$$

where  $C$  is the total switching capacitance,  $V_{dd}$  the voltage swing and  $f$  the switching frequency and the remaining part represents the leakage current of the transistor. Application Specific Integrated Circuits (ASICs) may also used to reduce the unnecessary power consumptions. In all these conditions , the design of sensor network security algorithms and protocols are influenced by the corresponding data processing and communication matrices.

## 2.3 WSN Protocol Stack

As seen from the architecture of WSN, it consists of several nodes scattered in the field . Every node collects the information, passes it more powerful node called as ‘sink or base station’. Base station sends information to the end user via satellite or internet. Current weather forecast on *Canberra metrological web site* is the classical example of WSN connected via internet. Sensor nodes and base station are governed by protocol stack as shown in the figure. The sensor network protocol stack [41] consists of 5 layers and three planes namely application layer, transport layer, network layer, data link layer, physical layer, power management plane, mobility management plane, and task management plane.

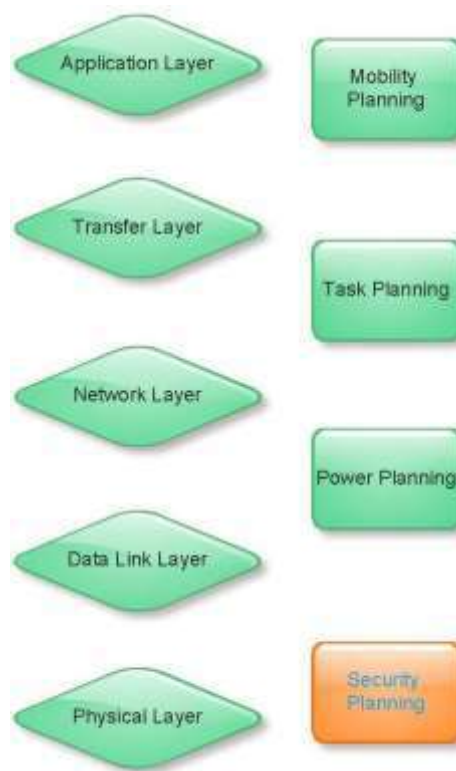


Figure 2. 5 WSN Protocol Stack with Proposed Security Plane

All the user interactive software forms the application layer. The next layer helps for the data flow and network layer guarantee the routing of data .The physical layers covers modulation, demodulation issues. Data link layer handles encryption, decryption, and error coding

matters. In addition to these layers there are three planes namely task, power and mobility. These planes assist sensor nodes to coordinate the sensing, power consumption and other issues. As the sensor nodes are deployed in hostile environment, security of the WSN is prime subject. Every layer in the protocol stack suffers from attacks from adversary. Instead of looking layer wise security, this research has proposed 'holistic approach'. In this type of approach instead of tackling layer wise attacks, a single approach is taken for all the layers. The reason behind this approach is, every lower layer performs specific task for the upper layer. For example routing layer route the information among the nodes but that can not be possible without modulation and demodulation which is performed by physical layer. So while making counter measures for routing attacks, physical layer must be taken in to consideration. This approach is called holistic approach. The more description about these attacks and countermeasures is available in the next subsections. While investigating ECC for WSN, we must take in to account the requirements of physical layer. The algorithms to be used must be selected according to the physical layer components like type of the microprocessor, receivers, available memory, packet size ,modulation and demodulation techniques, power consumption of transmitters and receives etc. This realistic approach has made algorithms proposed by this research readily accepted by industry.

## **2.4 WSN Topology**

WSN network consists of thousand of nodes .Some of them die due to battery problem or new one may be introduced to maintain the required density. This topic is of prime importance from security point of view. Some of the results obtained with SNetSim[51] are

hereby attached to give the idea about the deployment strategy of WSN. In this case study done by the author, number of sensors are 100 with radio frequency range of 200 meter. The width of the field is 1400 meter X 850 meter.

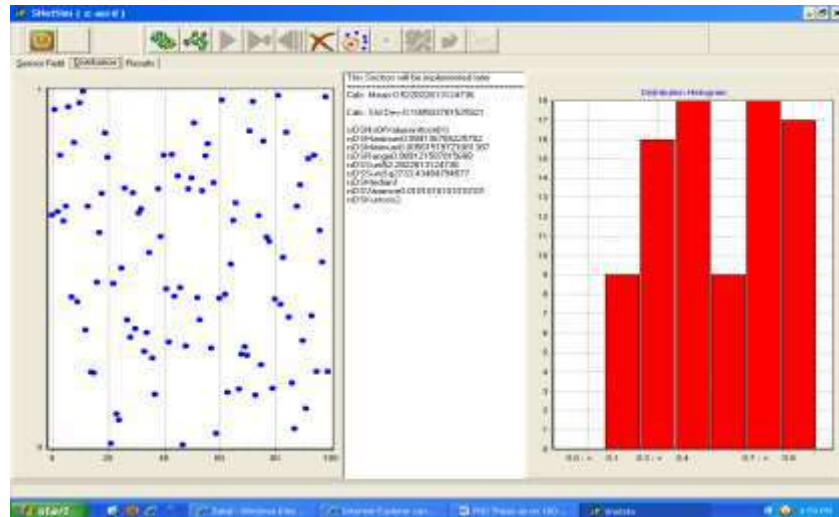


Figure 2. 6 Deployment of Nodes (Delphi Distribution)

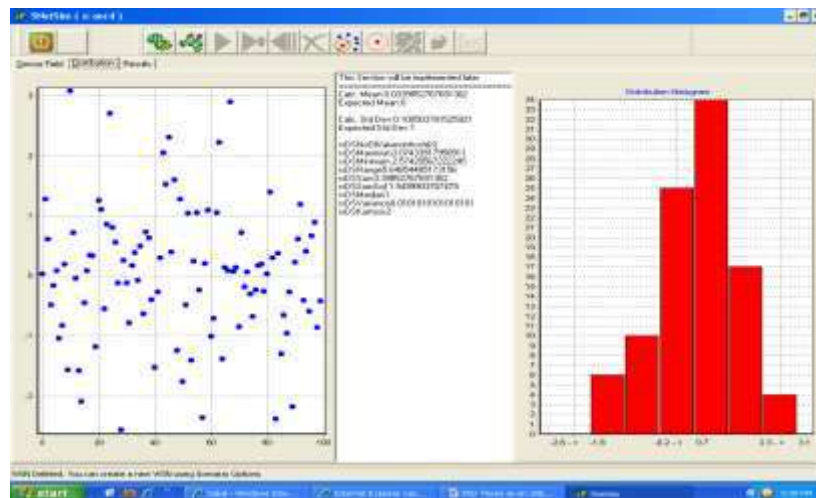
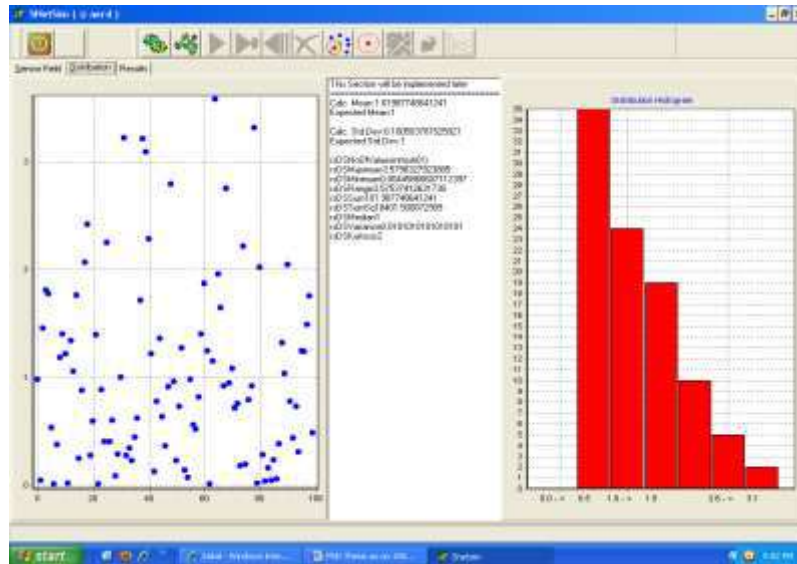


Figure 2. 7 Deployment of Nodes (Gaussian Distribution)



### Figure 2. 8 Deployment of Nodes (Exponential Distribution)

## 2.5 Network Layer Protocols Evaluation

To provide in sight in to how sensor nodes communicate with each other, following two popular schemes have been discussed-

**a. Flooding:** In flooding protocol[52] each and every node broadcast the packet unless a maximum number of hops for the packets have reached or the destination of the package is node itself. Flooding does not require any topology maintenance and is robust routing scheme. However it results in implosion where for example, if sensor node A has  $N$  neighbor sensor nodes that are also the neighbors of sensor node B, the sensor node B receives  $N$  copies of the message sent by sensor node A. It also results in duplicating of the messages as the two neighboring node sends the same information. Flooding protocol is having resource

blindness as it does not take in to account the energy sources and bandwidth of the network. The experimentation carried with SNetSim shows that for WSN of 100 nodes, flooding protocol resulted in 768 redundant packets. To avoid duplicate messages gossiping protocol provides better results.

**b. Gossiping:** In gossiping protocol[52] node searches for its nearest neighbor randomly and sends packet to it. The neighbor node in turns sends this packet to its neighbor. This simple approach avoids information implosion problem and number of redundant messages. But this protocol takes long time for execution. The figure shows the drastic reduction in the redundant count for gossiping protocol as compared to previous one.

## 2.6 Possible Attacks on WSN Illustrated Layer Wise

This section throws light on various attacks on WSN which are classified on the basis of protocol stack.

### 2.6.1. Physical Layer

**a. Denial of Service (DOS) attack:** Wood defines DoS attack as “any event that eliminates a network’s capacity to perform its expected function”[53]. Denial of service attacks on wireless sensor networks includes jamming the radio frequencies, violating MAC protocol, disabling the service or capturing the sensor nodes.

**b. Frequency Jamming:** This is one of the Denial of Service Attacks [54] in which the adversary attempts to disrupt the operation of the network by broadcasting a high-energy signal in the same frequency band in which WSN are operating (Center Frequency 916 MHz). [54] has classified Jamming attacks in WSNs, as *constant Jamming* which leads to corrupts packets as they are transmitted, *deceptive jamming* in which adversary sends a constant stream of bytes into the network to make it look like legitimate information, *random jamming or intermittent jamming* which alternates between sleep mode and active jamming mode, and *reactive jamming* which jam signal when it senses two nodes communicating.

Jamming can be avoided by use of Frequency spread spectrum or direct sequence spread spectrum. The jamming in network layer can be avoided by first mapping the jammed area and second routing information bypassing those areas.

**c. Radio Interferences:** These can be produced by use of oscillators which produces the same frequency in which WSN is operated. They can be avoided by use of symmetric key algorithms [55] like Micro TESLA in which the disclosure of the keys is delayed by some time interval.

**d. Tampering Node:** Enemy can capture the node as they are unattended and can retrieve the information such as cryptographic keys, node identification numbers or some sensitive data. These attacks can be prevented by use of tamper-proof the node's memory get vaporizes if somebody tampers physical package. The next attack is classical case of node capturing and analyzing power traces of microprocessor.

**d. Simple Power Analysis Attack:** In this type of attack[56, 57] information about the private key of the node is obtained by examining the power traces of single secret key operation. The hypothesis behind power analysis attack is that the power traces are correlated to the instructions the device is executing as well as the values of the operand.

For e.g. Let us assume microcontroller of the WSN node is executing instruction MVIA 11h, in which op code for MVIA is 3E and the operand is 11h. In binary format 3E number is recoded as 001111110. When there is 0 state, the CMOS transistor will be in cut off state and the whole VCC will appear at the output. On the other hand when the CMOS transistor will be in '1' state the transistor will be in saturation state and the output voltage will be zero. Therefore the examination of power traces can reveal the information about the instructions being executed and the contents of data registers. If the WSN node is executing the algorithm

for calculating the secret key, power traces may give hint for the secret key to the adversary. Consider for example WSN node is performing scalar point multiplication  $KP$  during ECDSA signature generation. Here  $P$  is a public key and  $k$  is a secret integer. Let us assume the binary method is used for calculating  $Q$ . From the power traces of Microcontroller it can be easily traced out the sequence of point doubling and point addition operations which will lead to actual value of integer  $K$  in binary form.

The following figures shows power traces obtained from Microprocessor.

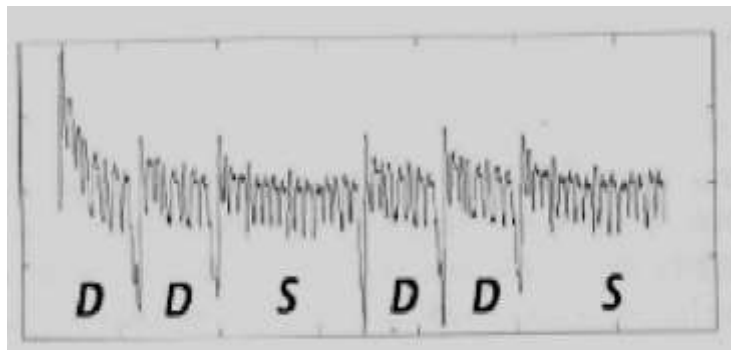


Figure 2. 9 Power Traces of SC140DS Processor

In this figure  $D$  represents point doubling operation and  $S$  represents point addition operation. When bit is zero point doubling is performed and when bit is 1 point addition is performed. So these power traces indicate that the value of  $K$  is 001001 which is actually secret key. The chapter 5 proposes a new algorithm for avoiding SPA attacks in ECC on WSN node. [58] Proposed algorithm known “double and add always” algorithm to prevent this type of attack.

This research proposes algorithm based on one’s complement subtraction for recoding of integer  $K$  as countermeasure for avoiding SPA attacks.

In the light of above facts it may be concluded that to design a secured protocol based on public key or private key which satisfies goals of confidentiality, message integrity, and data freshness is an open research issue. The above content indicates various attacks and requires different counter measures. This research will focus mainly on only eavesdropping, data alteration and false information injection attacks.



## 2.6.2 Data Link Layer

**a. Exhaustion:** A adversary node disrupts the MAC protocol, by continuously acquiring control of broadcast medium and transmitting over the medium denying access to other nodes for channel access which eventually leads a starvation for other nodes in the network [59] . This attack can be avoided by ignoring repeated request by node by channel acquisition, thus preventing energy drain caused by repeated transmissions. Other option is use of Time division multiplexing technique in which every node is given legitimate time slot for transmitting the information.

**b. Interrogation:** In this attack the adversary repeatedly send ‘request to send signal’ (RTS) to the targeted node to get Clear to send signal (CTS) to keep the targeted node busy and to exhaust its battery resources [59]. To avoid these type of attacks a node can limit RTS request from the same node or use Anti replay protection and strong link-layer authentication

**c. Sybil attack:** It is defined as ‘malicious device intelligently taking on multiple identities’. Sybil attack can create problem in routing information as well as data aggregation [60]. MAC protocols some time requires voting to know better path for transmission. Here the Sybil Attack could be used to stuff the ballot box as attackers take several identities and manipulate the fake voting.

## 2.6.3 Network Layer

**a. Sinkhole:** In a sinkhole attacks adversary’s node advertises high quality route to the base station and divert all the information flow through it and acts as *sinkhole*, prohibiting the information to reach the base station. A laptop class adversary with a powerful transmitter is the example of enemy node. Geo-routing protocols are resistant to sinkhole attacks, because

that topology is constructed using only localized information, and information is routed through physical location of the base station.

**b. Hello Flood:** In this attack lap top class adversary with power full transmitter flood the network with Hello message to pretend it as neighbor of all nodes. A node receiving such packets may assume that it is in radio range of the sender. This causes a large number of nodes sending packets to this imaginary neighbor and thus get mislead. These attacks can be avoided by doing authentication of all participating node by third party trusted certifying authority or by use of directionality of a link before taking action based on the information received over that link.

**c. Selective Forwarding:** In WSNs information is always communicated with single hop or multi hop fashion as the transmitting range of node is limited usually 10 meters and with assumption that all participating nodes with forward the information faithfully. In case of selective forwarding, malicious or attacking nodes refuse to pass information to the neighboring node. If it blocks all the information then it is referred as Black Hole Attack however if it selectively forward the messages, then it is called selective forwarding. To overcome this, Multi path routing can be used in combination with random selection of paths to destination, or braided paths can be used which represent paths which have no common link or which do not have two consecutive common nodes, or use implicit acknowledgments, which ensure that packets are forwarded as they were sent .

**d. Wormhole Attacks:** An adversary can dig messages received in one part of the network and replay them in another part of the network [61]. This is usually requires two adversary nodes, where the nodes try to understate their distance from each other, by broadcasting packets along an out-of-bound channel available only to the attacker. To overcome this, the information is routed to the base station along geographically shortest path or use very tight time synchronization among the nodes.

**e. Replayed Routing Information:** The most direct attack against a routing protocol in any network is to target the routing information itself while it is being exchanged between nodes.

An attacker may spoof, alter, or replay routing information in order to disrupt information flow in the network. These disruptions include the creation of routing loops, attracting or repelling network communication from select nodes, extending and shortening source routes, generating fake error messages, partitioning the network, and increasing end-to-end latency[9]. A countermeasure involves making use of a message authentication code (MAC). Efficient encryption and authentication techniques are useful to avoid these types of attacks.

**g. Misguiding path:** In this type of attacks adversary node advertises path that can route information in wrong direction through which the destination is unapproachable. In place of sending the packets in correct direction the attacker route the information to one targeted node and thus this node get flooded with any useful information. In this case if the node is getting plenty of messages without any useful information, then it is better to switch the node in sleeping condition.

**i. Homing:** This type of attacks targets mainly cluster heads that sends information to the base station or sink and is having special responsibilities of cryptographic task [59]. First of all cluster heads are identified followed by any denial of service attack like jamming or capturing these key network nodes. The countermeasures involve use of encryption scheme for header and use of mock-up packets throughout the network to equalize information volume. But this puts strain on bandwidth and other energy resources.

#### **2.6.4 Transmission Layer**

**De-synchronization Attacks:** In this attack, the adversary node requests for retransmission of missed frames purposefully and keeps authentic node busy in the futile activity. This causes wastage of energy and bandwidth of network in an end less synchronization-recovery

protocol. This situation can be prevented with proper encryption and authentication techniques of packets.

### 2.6.5 Application Layer

**a. Overwhelm attack:** In this attack adversary will give false stimuli to the sensor of the node to pretend new event and causes node to send bulk of information to the base station. This attack can be prevented by use of careful designing of the sensors and data-aggregation algorithms.

**b. Path-based DOS attack:** It involves injecting spurious or replayed packets into the network at leaf nodes. This attack can starve the network of legitimate traffic, because it consumes resources on the path to the base station, thus preventing other nodes from sending data to the base station. Combining packet authentication and anti replay protection prevents these attacks .

**c. Deluge (reprogram) attack:** Network-programming system let you remotely reprogram nodes in deployed networks If the reprogramming process isn't secure, an intruder can hijack this process and take control of large portions of a network. It can use authentication streams to secure the reprogramming process .

## 2.7 A survey of cryptography protocols for WSN

All above discussion underlines the requirement of various cryptographic operations, including encryption, decryption and authentication for reliable operation of WSN. Selecting the appropriate cryptography algorithm for WSN is fundamental to providing security services however, the decision depends on the computation and communication capability of the sensor nodes. Applying any cryptographic algorithm requires transmission of extra bits, extra processing, memory and battery power, which are very important resources for the sensors longevity. The process by which cryptographic algorithm should be selected is based on the following criteria [59],

1. **Energy:** how much energy is required to execute the encrypt/decryption functions
2. **Program memory:** the memory required to store the encryption/decryption program
3. **Temporary memory:** the required RAM size or number of registers required temporarily when the encryption/decryption code is being executed
4. **Execution time:** the time required to execute the encryption/decryption code.
5. **Program parameters memory:** the required memory size to save the required number of keys used by the encryption/decryption function.

The next paragraphs will give very brief survey of the contemporary security protocols for WSN. Cryptographic protocols are broadly classified into two major types namely *symmetric key* and *asymmetric key or public key protocols*. In most of the cases, standard assumption has been made that WSN node will be unable to support traditional computational intensive public key cryptography [15]. Early attempt to implement RSA algorithms on motes did not meet with success. Because of this many traditional secure schemes that employ public key approaches were deemed not suitable for sensor networks. However recent advances have shown that PKC is feasible on recent nodes [16]. The following section gives brief overview of TinySec [62] and SPINS [63] protocols which are the primitive protocols for understanding of WSN security and are worth to discuss.

### **2.7.1 SPINS: Security Protocol for Sensor Network**

Due to its resource constrained nature, cryptography is the best approach for WSN security. [63] established “SPINS”, a suite of cryptographic protocol for the first generations of motes

which were highly deficient in bandwidth, memory and computational power. SPINS consists of two parts namely SNEP and  $\mu$ TESLA. SNEP takes care of data confidentiality, authentication of parties, and data freshness.  $\mu$ TESLA ensures authentic broadcast in a harsh environment.

**a. SNEP:** It is a secure mechanism that efficiently provides end to end data confidentiality, integrity, authenticity and freshness for unicast messages between two nodes, possibly separated by multiple hops. Every packet is encrypted with a stream cipher based on a symmetric key operating in block cipher counter mode [64]. Every packet also contains message authentication code. A counter is used in message authentication code to prevent the reply attacks. To save bandwidth, the counter is not sent with the message. The receiver synchronises the counter with the sender by receiving a packet. If some packets are lost, sender and receiver need to resynchronise the counter for packets.

**b.  $\mu$ TESLA :** The basic principle of the  $\mu$ TESLA is ‘to achieve asymmetric cryptography by delaying the disclosure of symmetric keys [63]’. The base station broadcasts a message with message authentication code generated with a symmetric key. The symmetric key is one-way hash chain. To securely broadcast a message the base station first sends the message and its message authentication code generated by a key. At a later stage when the receiving node receives the message, the base station releases the key according to its delayed key disclosure schedule. When the node receives the key it verifies whether key is on their one way hash function. If verified then the message authentication code is tested as to whether it was generated by the base station. Figure 4 explains the operation in more detail.

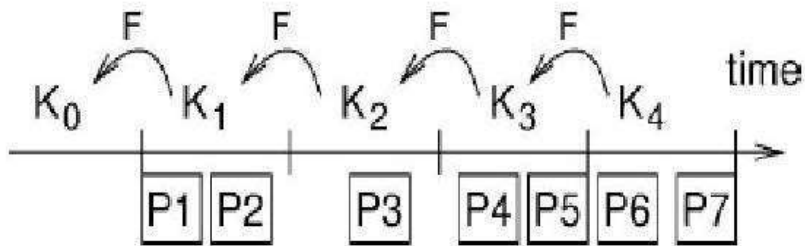


Figure 2. 10 $\mu$ TESLA One Way Key Function[63]

One problem associated with  $\mu$ TESLA is that when many broadcast messages are lost, the sensor node has to spend a long time repeatedly executing a one way hash function to verify the key. Another issue in  $\mu$ TESLA is delayed message authentication. When the sensor node receives the broadcast message, it can not verify the message until base station releases key of the message authentication code several time slots later. In the intervening time, an adversaries can launch DoS attacks, flooding the network with unverifiable packets[64].

**2.7.2 TinySec:** [62] demonstrated TinySec, the first fully implemented protocol for link layer cryptography in sensor networks. TinySec makes use of Tiny OS 1.0. It offers message integrity, confidentiality, authenticity for two neighbouring nodes exchanging the information. It also gives prime importance to low computation, lightweight memory use, low communication overheads and last, but not the least, minimum energy consumption. The TinySec mechanism is divided into two parts: authenticated encryption (TinySec –AE) and authentication only (TinySec –Auth). Tiny Sec uses four byte message authentication code to protect the integrity and authenticity of the message. It also uses Cipher Block Chaining

(CBC) mode for encryption, with an initialization vector based on a two byte counter and the header of the TinyOS packet [64].

## **2.8. Summary**

1. Recent studies on public key cryptography have demonstrated that public key operations may be practical in sensor networks. However, calculating key is expensive in terms of computation and energy cost to accomplish in a sensor node. The methods to accelerate key calculation time of sensor nodes needs to be studied further.
2. Symmetric key cryptography is superior to public key cryptography in terms of speed and low energy cost. However, the key distribution schemes based on symmetric key cryptography are not perfect.
3. Most current symmetric key schemes for WSNs aim at link layer security for one-hop communications, but not upper layer security for multi hop communications, because usually, it is unlikely for each node to store a transport layer key for each of the other nodes in a network due to the huge number of nodes. A more promising approach is to combine both symmetric and asymmetric cryptography techniques.
4. Proving the authenticity of public keys is another important problem. (To overcome this problem this research has proposed a innovative algorithm to avoid man in the middle attack)
5. Key revocation is another un-addressed problem. It is very difficult to design a universal key revocation scheme. It is still an open problem for resource constrained WSNs.
6. Current proposed key management schemes assume that the base station is trustworthy. However, there may be situations (e.g., in the battlefield) where enemy can destroy the base station.
7. All key management protocols discussed in literature so far are based on symmetric key cryptography. Key management schemes based on public key cryptography need to be



designed. A typical WSN may contain from hundreds to thousands of sensor nodes. Any protocol used for key management and distribution must be adaptable to such scales.

8. For any pair of nodes that do not share a key and are connected by multiple hops, there needs to be assigned a path-key to guarantee end-to-end secure communication. Such a path key establishment needs to be improved

9. Another challenging issue is that each node needs to discover a neighbor in wireless communication range with which it shares at least one key. A good-shared key discovery approach should not permit an attacker to know shared keys between every two nodes.

10. Most key management schemes discussed in literature so far are suitable for static WSNs. Following technique advance, key management and security mechanisms for mobile WSNs should be considered and become a focus of attention.

11. Though many key management approaches consider defending against node compromise, the efficiency and security performance is not high when their mechanisms are deployed in some special application environment.

Thus, the above discussion underlines the need of new research in public key cryptography for WSN which is a promising area.

## Chapter 3 Elliptical Curve Cryptography Modelling

---

### 3.1 Introduction

Elliptical curves have been studied in mathematics for along time, but their use in cryptographic applications was first suggested by Neal Kibitz [7] and Victor Miller [65]. About 25 years of active investigation have confirmed the beneficial properties of these systems and led to the invention of efficient implementations methods. In the 21<sup>st</sup> century the use of ECC for production of cryptographic primitives, such as digital signatures has begun to put in to standards. We have now, for instance, the US standards ANSI X9.62, FIPS 186-2 NIST, SECG, IEEE P1363.

While implementing an ECC an important consideration is how to implement the underlying field arithmetic. The problems encountered in such implementations are addressed in this chapter, with attention being focused on questions which arises mostly in software implementations, although some hardware issues are briefly mentioned. Two questions of particular importance are, whether to use even or odd characteristics fields and secondly, whether to restrict the implementation to fields of a special type, for efficiency in case of WSN.

### 3.2 Definition of Elliptical Curve

An elliptic curve  $E$  over  $GF(p)$  where  $p$  is a prime and  $p > 3$  is defined as the points  $(x, y)$  satisfying the curve equation  $E: y^2 \equiv x^3 + ax + b \pmod{p}$  where  $a$  and  $b$  are constants satisfying  $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$ . Points satisfying this equation are known as *affine points*. In addition to these points a point at infinity  $\infty$  is also said to be on the curve. The set of all points on the curve  $E$  is denoted by  $\neq E(GF(p))$  and is called order of the curve. The example 3.1 shows an elliptical curve and its group elements.

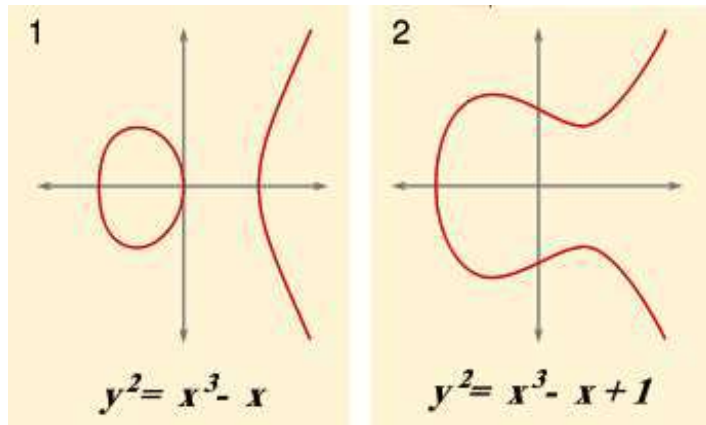


Figure 3. 1 An Elliptical Curve Example

Table 3. 1 Nomenclatures used in ECC

| Nomenclatures   |
|---|
| <p><math>E</math> : An elliptic curve over <math>GF(p)</math> where <math>p</math> is a prime and <math>p &gt; 3</math> , defined as the points <math>(x, y)</math> satisfying the curve equation <math>E : y^2 \equiv x^3 + ax + b \pmod{p}</math> where <math>a</math> and <math>b</math> are constants satisfying <math>4a^3 + 27b^2 \not\equiv 0 \pmod{p}</math>.</p> |
| <p><math>p</math> : A large prime which specifies the field over which the elliptical curve is defined, <math>GF(p)</math>.</p>   |
| <p><math>a, b</math> : Constant curve parameters satisfying equation as above.</p>  |
| <p><math>x, y</math> : The <math>x</math> and <math>y</math> coordinates of a point on the curve in affine system.</p>  |
| <p><math>G</math> : A Base point on the curve with order <math>n</math> .</p>   |
| <p><math>P, Q, R</math> : Points on the Elliptical Curve.</p>   |
| <p><math>\#E(GF(p))</math> or <math>\eta</math> : The order of the curve (The total number of points on the curve)</p>  |
| <p><math>\infty</math> : Point at infinity.</p>   |
| <p><math>d</math> : Private key of node</p>   |

All the points on the elliptical curve forms a finite *abelian group* meaning that if any two points on the curve are added , their sum point is on the curve itself forming a closed cyclic group with point at infinity  $\infty$  serving as identity element.

---

### Example 3.1 Elliptical Curve and its group elements

---

Let  $E$  is an elliptical curve with  $p = 41, a = 2, b = 4$  and defining equation

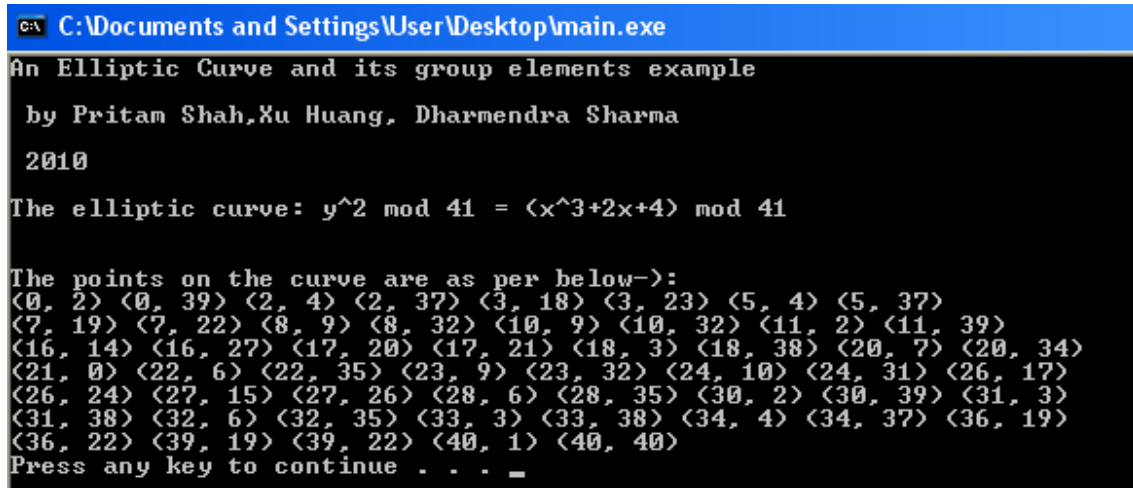
$$y^2 \equiv x^3 + 2x + 4 \pmod{41}$$

Then the points on  $E$  are-

(0, 2) (0, 39) (2, 4) (2, 37) (3, 18) (3, 23) (5, 4) (5, 37) (7, 19) (7, 22) (8, 9) (8, 32) (10, 9)  
(10, 32) (11, 2) (11, 39) (16, 14) (16, 27) (17, 20) (17, 21) (18, 3) (18, 38) (20, 7) (20, 34)  
(21, 0) (22, 6) (22, 35) (23, 9) (23, 32) (24, 10) (24, 31) (26, 17) (26, 24) (27, 15) (27, 26)  
(28, 6) (28, 35) (30, 2) (30, 39) (31, 3) (31, 38) (32, 6) (32, 35) (33, 3) (33, 38) (34, 4)  
(34, 37) (36, 19) (36, 22) (39, 19) (39, 22) (40, 1) (40, 40).

---

\*The C++ code for generating group elements of elliptical curve for above example is given in Appendix C. The actual output window is given below-



```
C:\Documents and Settings\User\Desktop\main.exe
An Elliptic Curve and its group elements example
by Pritam Shah, Xu Huang, Dharmendra Sharma
2010
The elliptic curve:  $y^2 \pmod{41} = (x^3 + 2x + 4) \pmod{41}$ 
The points on the curve are as per below-:
(0, 2) (0, 39) (2, 4) (2, 37) (3, 18) (3, 23) (5, 4) (5, 37)
(7, 19) (7, 22) (8, 9) (8, 32) (10, 9) (10, 32) (11, 2) (11, 39)
(16, 14) (16, 27) (17, 20) (17, 21) (18, 3) (18, 38) (20, 7) (20, 34)
(21, 0) (22, 6) (22, 35) (23, 9) (23, 32) (24, 10) (24, 31) (26, 17)
(26, 24) (27, 15) (27, 26) (28, 6) (28, 35) (30, 2) (30, 39) (31, 3)
(31, 38) (32, 6) (32, 35) (33, 3) (33, 38) (34, 4) (34, 37) (36, 19)
(36, 22) (39, 19) (39, 22) (40, 1) (40, 40)
Press any key to continue . . .
```

Figure 3. 2 An Elliptical Curve and Its Group Elements

### 3.2 Pyramid of Elliptical Curve Cryptography (ECC)

The ECC pyramid in Figure 3.3 shows the hierarchies of the operations involved in implementing ECC on WSN platform. The ECC pyramid has three levels of arithmetic. The foundation of pyramid is the field or modular arithmetic. The field arithmetic is used to implement elliptical curve point addition and doubling operations. The point doubling and additions are the basic building blocks of the scalar multiplication process. All standard cryptographic protocols involve use of scalar multiplication which is the most time consuming operation of ECC. To reduce this time for WSN was main objective of this thesis. The ECC operations pyramid forms the basis of this chapter. In analyzing of ECC, top to bottom approach has been taken. Initially we will discuss standard cryptographic protocols such as ECDH [22], ElGamal encryption and decryption schemes[66], Elliptical Curve Digital Signature Algorithm (ECDSA) [67-69] which sit on the top of pyramid. It will be followed by scalar multiplication which in turns depends on EC point doubling and point addition operations. The chapter will end with brief review of finite field arithmetic algorithms available with their merits and demerits.

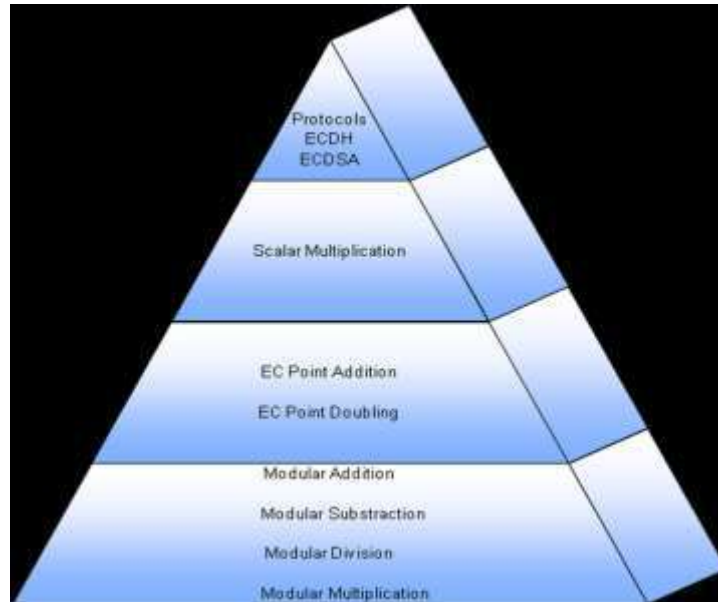


Figure 3. 3 ECC Operational Pyramid

### 3.3 Elliptic Curve Protocols for WSN

ECC can be used in WSN to obtain authentication of sensor nodes, maintaining data confidentiality, data integrity and data freshness. There are several standard protocols available in literature which can be tailored as per resources of WSN to achieve above goals. The next sections provide mathematical modelling for the same.

#### 3.3.1 Elliptic curve key generation

Let  $E$  be an elliptical curve defined over a finite field  $GF(p)$ . Let  $G$  be a point in  $\#E(GF(p))$  and has order  $n$ . Then the cyclic subgroup of  $\#E(GF(p))$  generated by

$G$  is,

$$\langle G \rangle = \{\infty, G, 2G, 3G, \dots, (N-1)G\}.$$

The  $(p, E, G, n)$  are the public domain parameters. A private key is an integer  $d$  that is selected randomly from the interval  $(1, n-1)$ , and the corresponding public key is  $Q = dG$ . Algorithm 3.1 shows the process to calculate public key in ECC. The problem of determining  $d$  given the domain parameters and  $Q$  is the elliptic curve discrete logarithmic problem (ECDLP).

**Algorithm 3.1 Elliptic Curve key pair generation**

---

Algorithm 3.1 Elliptic Curve key pair generation

---

Input: Elliptic Curve domain parameters  $(p, E, G, n)$ .

Output: Public key  $Q$  and private key  $d$ .

1. Select  $d \in (1, n-1)$ .
  2. Compute  $Q = dG$ .
  3. Return  $(Q, d)$ .
- 

### 3.3.2 Definition of Elliptic Curve Discrete Logarithm Problem (ECDLP)

Given a prime modulus  $p$ , curve constants  $a$  and  $b$ , two points  $G$  and  $Q$  on the corresponding curve, the elliptic curve discrete logarithm problem (ECDLP) is to find a scalar  $d$  such that  $Q = [d]G$ . The value  $d$  is known as the elliptic curve discrete logarithm (ECDL) [57].

The elliptical curve is said to be secure for cryptography if the number of points on the curve  $\#E(GF(p))$  which is also called as order of the curve should be as large as possible. In



practice Computer multiplication (CM) method is used to generate the secure curve and till date there is no attack reported on the curve generated by this method. Other method involves use of a subfield curve also known as Koblitz Type. These curves are generated by choosing a small field referred as Galois field. There are curves over composite extension field  $GF(p^m)$  where  $m$  is not prime but are considered less secure as compared to  $GF(p), GF(2^r)$  where  $p, r$  are prime due to the method of Weil descent for solving the ECDLP. The next method involves use of randomly chosen values of  $p, a, b$  and to use Schoof-Elkies –Atkin algorithm to count the number of points on the curve. Unfortunately this algorithm is very time consuming in spite of fact that they are much secured. For example, one implementation takes several minutes on a Intel Pentium 1.80 GHz computer with 768 RAM to count the points on 160 bit and 192 bits curve which prohibits use of them for resource constraint WSN. Due to this fact this research recommends use of fixed curves published by national Institute of Standards and Technology ( NIST) [70] for WSN. The list of NIST curves is attached as an Appendix A with this thesis. WSN security aspects of authenticity, message integrity, and confidentiality can be achieved with the help of following standard protocols which are based on elliptic curve.

### **3.3.3 Elliptical Curve Diffie-Hellman Protocol (ECDH)**

ECDH[22] is a protocol which allows Alice and Bob to establish a shared secret key for communication over a unsecured channel in which Eve is eavesdropper. The shared secret key can be used to encrypt the subsequent communication using symmetric key cipher.

---

**Algorithm 3.2 Elliptical Curve Diffie-Hellman Protocol**

---

1. Alice and Bob agree on the elliptic curve  $E$  and base point  $G(x_1, y_1)$ .
  2. Alice generates a random integer  $a \in \{1, \dots, n-1\}$  where  $n$  is the order of the group and number  $a$  is called private key of the Alice.
  3. Alice sends to Bob her public key  $Q_a = aG = a(x_1, y_1) = (x_a, y_a)$ .
  4. Bob generates the random integer  $b \in \{1, \dots, n-1\}$  and number  $b$  is called private key of the Bob.
  5. Bob sends to Alice his public key  $Q_b = bG = B(x_1, y_1) = (x_b, y_b)$ .
  6. Alice can then compute  $(x_k, y_k) = aQ_b = a(bG) = abG$ .
  7. Likewise, Bob can compute  $(x_k, y_k) = bQ_a = b(aG) = abG$ .
  8. The shared session key is  $x_k$  which the  $x$ -coordinate of the point.
- 

The protocol is secure because no parties know the private key of the Alice and Bob. To get this private key, they need to solve ECDLP. The only information that Eve knows is  $G, E, Q_a, Q_b$ . The major problem for this protocol is the ‘man in the middle attack’ which has been discussed in the chapter 8. In this type of attack, the Eve sends public key to Alice and as Alice is not sure about Eves identity establishes the session key with him. This problem can be solved by using a certifying authority for all public keys.

The security of the this algorithm depends on the length of the key in bits. To attack an algorithm with  $k$  bits key, generally  $2^k - 1$  operations are required. Hence the parameters of

the curve  $E$  have to be chosen in such a way that at least  $2k - 1$  operations will be required to break the key. The table 3.1 gives recommended key sizes by National Institute of standard and Technology (NIST) [70] to protect the public key for DES, AES, RSA and ECDH encryption methods.

**Table 3. 2 NIST Guidelines for Key Sizes**

| Symmetric Key<br>Sizes in Bits | Symmetric Encryption<br>Algorithm | Minimum Size Bits of Public Keys |       |     |
|--------------------------------|-----------------------------------|----------------------------------|-------|-----|
|                                |                                   | DSA/DH                           | RSA   | ECC |
| 80                             | Skipjack                          | 1024                             | 1024  | 160 |
| 112                            | 3DES                              | 2048                             | 2048  | 224 |
| 128                            | AES-128                           | 3072                             | 3072  | 256 |
| 192                            | AES-192                           | 7680                             | 7680  | 384 |
| 256                            | AES-256                           | 15360                            | 15360 | 512 |

One will notice from above table, as the bit sizes in symmetric key increases, the bit sizes in DSA and RSA[6] increases at much faster rate than ECC. This is the reason why ECC offers more security per bit as compared to RSA and DSA algorithms.

In choosing Elliptical Curve for cryptography there are various options available. The NIST has recommended 15 standard curves with various sizes. Ten of these curves are for binary field and five are for prime field. The list of these curves is given in Appendix A of the thesis.

### **3.3.4 Comparison between RSA and ECC for WSN**

In PKC there are two options available namely, ‘factorial based’ RSA algorithm (1024 bits) and ‘discrete logarithmic problem’ based ECC (168 bits). The ECC based Diffie-Hellman is more suitable for WSN than RSA based DH as the former one provides compact keys and signatures. The compact key of ECDH will help to reduce computational cost, battery power, and bandwidth of the network. The broadcast authentication can be achieved with ECDSA scheme without requiring time synchronization. Due to all these reasons, this research will investigate the application of ECC for resource constraint WSN platforms.

ECC provides flexibility to wireless sensor node without compromising the security of the whole network as the key are not pre distributed. There is also no need of time synchronization which is very difficult task. The example 3.2 below compares key sizes of ECC and RSA.

---

### Example 3.2 Comparing key sizes of RSA and ECDH on MIRACL crypto library.

---

An Elliptic Curve cryptography example by Pritam Shah, Xu Huang, and Dharmendra Sharma 2010 executed on MIRACL crypto Library

First Diffie-Hellman Key exchange...

Alice's offline calculation

Bob's offline calculation

Alice calculates Key=

```
9663095517842289273904410014104171454484284808865248581166028165813424331850733279027543004977467106972723747313
7783726663658673300717817399217987732570637229960484394094366010660167666439045332982888866757555461807181449167
71753175560841094136256355000940025169383417651031434752564647309718507822261318989
```

Bob calculates Key=

```
9663095517842289273904410014104171454484284808865248581166028165813424331850733279027543004977467106972723747313
7783726663658673300717817399217987732570637229960484394094366010660167666439045332982888866757555461807181449167
71753175560841094136256355000940025169383417651031434752564647309718507822261318989
```

Alice and Bob's keys should be the same! Let's try that again using elliptic curves....

Alice's offline calculation

Bob's offline calculation

Alice calculates Key=

```
5756659004977655693910975295747305450313129196486344767877
```

Bob calculates Key=

```
5756659004977655693910975295747305450313129196486344767877
```

Alice and Bob's keys should be the same! (But much smaller)

\* Code for the above program is given in Appendix A. The experiment carried out on MIRACL crypto library.

```

C:\WINDOWS\system32\cmd.exe
First Diffie-Hellman Key exchange ....
Alice's offline calculation
Bob's offline calculation
Alice calculates Key=
96630955178422892739044100141041714544842848088652485811660281658134243318507332
79027543004977467106972723747313778372666365867330071781739921798773257063722996
04843940943660106601676664390453329828888667575554618071814491677175317556084109
4136256355000940025169383417651031434752564647309718507822261318989
Bob calculates Key=
96630955178422892739044100141041714544842848088652485811660281658134243318507332
79027543004977467106972723747313778372666365867330071781739921798773257063722996
04843940943660106601676664390453329828888667575554618071814491677175317556084109
4136256355000940025169383417651031434752564647309718507822261318989
Alice and Bob's keys should be the same!

Lets try that again using elliptic curves ....
Alice's offline calculation
Bob's offline calculation
Alice calculates Key=
5756659004977655693910975295747305450313129196486344767877
Bob calculates Key=
5756659004977655693910975295747305450313129196486344767877
Alice and Bob's keys should be the same! (but much smaller)

```

Figure 3. 4 Diffie Hellman Key Exchange Protocol

### 3.3.5 ElGamal Elliptic Curve Protocol

In this scheme a plain text message  $m$  is first represented as a point  $M$  on the curve, then encrypted by adding it to  $kQ$  where  $k$  is randomly selected integer, and  $Q$  is the intended receiving nodes public key. The sender sends the point  $C_1 = kG$  and  $C_2 = M + kQ$  to the receipts who uses her private key  $d$  to compute the original message.

#### Algorithm 3. 3 Basic ElGamal Encryption Algorithm

---

##### Algorithm 3.3 Basic ElGamal Encryption Algorithm

---

Input: Elliptic curve domain parameters  $(p, E, G, n)$ , public key of receiver  $Q$ , plain text  $m$ .

Output: Cipher text  $(C_1, C_2)$ .

1. Represent the message  $m$  as a point  $M$  in  $E$ .
2. Select  $k \in \{1, \dots, n-1\}$ .
3. Compute  $C_1 = kG$ .
4. Compute  $C_2 = M + kQ$ .
5. Return  $(C_1, C_2)$ .

---

#### Algorithm 3. 4 Basic ElGamal Decryption Algorithm

---

##### Algorithm 3.4 Basic ElGamal Decryption Algorithm

---

Input: Elliptic curve domain parameters  $(p, E, G, n)$ , private key public key  $d$ ,

Cipher text  $(C_1, C_2)$

Output: Plain Text Message  $m$ .

1. Compute  $M = C_2 - dC_1 = (M + kQ) - d(kG) = (M + kQ) - k(dG) = M + kQ - kQ = M$ .
  2. Extract  $m$  from  $M$ .
-

To recover the message  $m$ , an eavesdropper needs to compute  $kQ$  which is called ECDLP problem. To make this scheme more clear to understand an example based on ElGamal scheme based on Elliptic Curve is given here for encryption and decryption of the message. In WSN every node sends the valuable information to the base station like temperature, pressure, humidity or more sensitive information. That information can be encrypted with the help of above protocol.

---

### Example 3.3 Basic ElGamal Encryption and Decryption based on Elliptic Curve

---

An Elliptic Curve cryptography example by Pritam Shah 2010.

The elliptic curve over finite field and having equation:

$y^2 \text{ mod}(263) = x^3 + 1x + 1(\text{mod}263)$  The points on the curve which satisfy the above equation are,

(0, 1) (0, 262) (1, 23) (1, 240) (2, 96) (2, 167) (3, 89) (3, 174) (4, 73) (4, 190) (6, 111) (6, 152) (7, 80) (7, 183) (8, 28) (8, 235) (10, 119) (10, 144) (14, 38) (14, 225) (15, 32) (15, 231) (21, 128) (21, 135) (22, 64) (22, 199) (23, 57) (23, 206) (24, 35) (24, 228) (27, 81) (27, 182) (29, 111) (29, 152) (30, 87) (30, 176) (31, 34) (31, 229) (33, 27) (33, 236) (38, 101) (38, 162) (39, 45) (39, 218) (40, 55) (40, 208) (44, 65) (44, 198) (45, 35) (45, 228) (47, 81) (47, 182) (48, 60) (48, 203) (49, 123) (49, 140) (51, 64) (51, 199) (57, 25) (57, 238) (58, 5) (58, 258) (59, 6) (59, 257) (60, 123) (60, 140) (61, 52) (61, 211) (66, 34) (66, 229) (67, 119) (67, 144) (68, 74) (68, 189) (69, 108) (69, 155) (72, 85) (72, 178) (74, 4) (74, 259) (75, 25) (75, 238) (77, 116) (77, 147) (78, 53) (78, 210) (81, 0) (82, 27) (82, 236) (83, 114) (83, 149) (87, 61) (87, 202) (88, 91) (88, 172) (99, 79) (99, 184) (103, 131) (103, 132) (108, 83) (108, 180) (110, 130) (110, 133) (111, 77) (111, 186) (112, 44) (112, 219) (115, 59) (115, 204) (116, 125) (116, 138) (117, 18) (117, 245) (118, 106) (118, 157) (123, 23) (123, 240) (127, 2) (127, 261) (131, 25) (131, 238) (132, 70) (132, 193) (134, 29) (134, 234) (137, 107) (137, 156) (138, 29) (138, 234) (139, 23) (139, 240) (141, 109) (141, 154) (142, 26) (142, 237) (143, 37) (143, 226) (144, 69) (144, 194) (145, 115) (145, 148) (146, 97) (146, 166) (147, 94) (147, 169) (148, 27) (148, 236) (150, 129) (150, 134) (152, 124) (152, 139) (154, 123) (154, 140) (157, 100) (157, 163) (159, 102) (159, 161) (162, 104) (162, 159) (166, 34) (166, 229) (169, 107) (169, 156) (170, 130) (170, 133) (173, 90) (173, 173) (174, 109) (174, 154) (175, 20) (175, 243) (180, 122) (180, 141) (181, 59) (181, 204) (182, 110) (182, 153) (184, 43) (184, 220) (185, 127) (185, 136) (186, 119) (186, 144) (188, 70) (188, 193) (189, 81) (189, 182) (190, 64) (190, 199) (192, 15) (192, 248) (194, 35) (194, 228) (195, 7) (195, 256) (196, 116) (196, 147) (199, 2) (199, 261) (200, 2) (200, 261) (206, 70) (206, 193) (207, 117) (207, 146) (208, 24) (208, 239) (210, 79) (210, 184) (211, 109) (211, 154) (216, 4) (216, 259) (217, 79) (217, 184) (218, 108) (218, 155) (219, 118) (219, 145) (220, 107) (220, 156) (221, 33) (221, 230) (222, 58) (222, 205) (223, 50) (223, 213) (224, 9) (224, 254) (226, 99) (226, 164) (228, 111) (228, 152) (230, 59) (230, 204) (236, 4) (236, 259)



(239, 108) (239, 155) (240, 31) (240, 232) (242, 72) (242, 191) (245, 48) (245, 215) (246, 130) (246, 133) (249, 98) (249, 165) (251, 75)  
 (251, 188) (253, 116) (253, 147) (254, 29) (254, 234) (259, 14) (259, 249) (260, 51) (260, 212).

### Basic ElGamal Message Encryption Scheme based on above elliptical curve

---

Let us choose base point from above group of elements for encryption of message ( $G$  is shown in green fonts and underlined),

$G = (148, 27)$  and order of  $G$  is 15.

Alice's public key

$$P_a = a * G = 51 * (148, 27) = (3, 89)$$

Bob's public key

$$P_b = b * G = 212 * (148, 27) = (61, 52)$$

Eve's public key

$$P_e = e * G = 153 * (148, 27) = (195, 256)$$

Plain text message from Alice to Bob

$$(m_1, m_2) = (19, 72).$$

Calculate

$$P_a = a * G, C_1 = a * P_b * m_1, C_2 = a * P_b * m_2.$$

Encrypted message from Alice to Bob

$$(P_a, C_1, C_2) = \{(3, 89), 193, 153\}.$$

Bob's decrypted message from Alice

$$m_1 = C_1 / (b * P_a), m_2 = C_2 / (b * P_a) = (19, 72).$$

Eve's decrypted message from Alice = (203, 191).

---

### 3.4 Point Addition and Point Doubling on Elliptical Curve

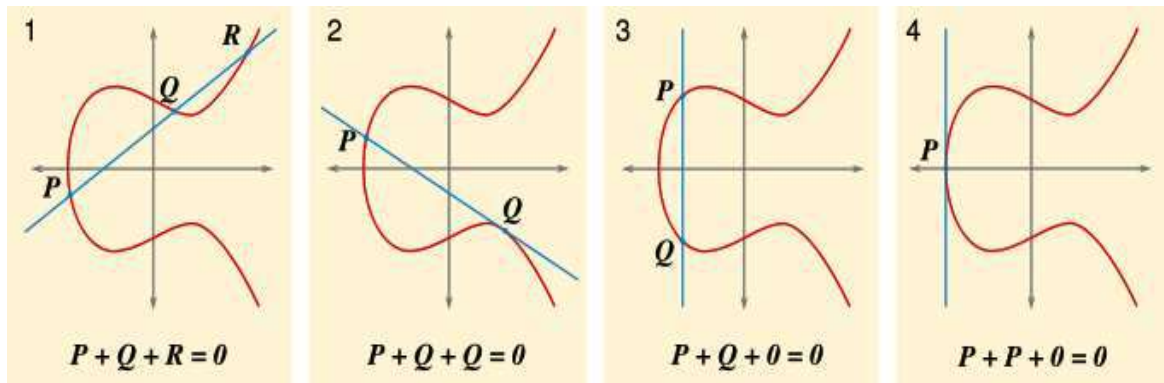


Figure 3. 5 Point Addition, Point Doubling Operations on Elliptical Curves

The second level of ECC pyramid comprises of point addition and point doubling operations.

Scalar multiplication of a point  $P$  by a scalar  $k$  is defined as :

$$[k]P = P + P + \dots + P \quad \dots k \text{ times}$$

This in turn consists of point doubling and addition operations. The following figure shows geometrical representations of point addition and doubling over the elliptical curve having equation  $y^2 = x^3 - 5x + 7$ .

**a. Point doubling:** To double a point  $P$ , a tangent to the curve at the point  $P$  is drawn, which intersects the curve at  $-R$ . A mirror point of  $-R$  is  $R$ , which is taken to be the result of the doubling operation.

**b. Point addition:** To add two points P and Q, a line is drawn through the two points and the line intersects the curve at a place, -R. A mirror point of -R is R, which is taken to be the result of the addition operation.

**c. Point at infinity:** If the two points P,Q are lying on vertical line, there addition is considered as point at infinity. Algorithms 3.5 and 3.6 describe point addition and point doubling respectively.

#### Algorithm 3.5 Point Addition

---

#### Algorithm 3.5 Point Addition

---

- Let P and Q be the two points on the elliptical curve .
- If P is  $\infty$  then return Q as the result.
- Else if Q is  $\infty$  then return P as the result.
- Else:
  - Let  $P = (x_1, y_1)$  and  $Q = (x_2, y_2)$ .
  - Let  $T_1 = (y_2 - y_1)$
  - Let  $T_2 = (x_2 - x_1)$
  - If  $T_2$  is zero then
    - \* If  $T_1$  is zero then return Double (Q) as the result.
    - \* Else return  $\infty$  as the result.
  - Let  $\lambda = T_1 \cdot T_2^{-1}$
  - Let  $x_3 = \lambda^2 - x_1 - x_2$

- Let  $y_3 = \lambda(x_1 - x_3) - y_1$
- Return  $(x_3, y_3)$ .

#### Algorithm 3. 6 Point Doubling

#### Algorithm 3.6 Point Doubling

- Let the point to be doubled be P.
- If P is  $\infty$ , return  $\infty$  as the result.
- Else
  - Let  $P = (x_1, y_1)$ .
  - Let  $\lambda = (3x_1^2 + a) \cdot (2y_1)^{-1}$
  - Let  $x_3 = \lambda^2 - 2x_1$
  - Let  $y_3 = \lambda(x_1 - x_3) - y_1$
  - Return  $(x_3, y_3)$ .

### 3.5 Finite Field Arithmetic

The efficient implementation of finite field arithmetic is an important prerequisite of elliptical curve cryptography on WSN because curve operations are performed using arithmetic operations in the underlying field. Three kinds of fields those are available for efficient implementation namely

1. Prime field
2. Binary field and
3. Optimal extension field.

For efficient implementation of finite arithmetic two basic operations are necessary modular reduction and modular inversion.

**a. Definition of Modular Reduction:** Reducing a number  $x \pmod{p}$  can be defined as finding the remainder  $r$  when  $x$  is divided by  $p$ .

Without modular reduction, the size of the numbers will become substantially larger, and thus addition and multiplication will take larger amounts of time and require the use of more memory.

Another operation which is required by an ECC is modular inversion.

**b. Definition of Modular Inversion:** The inverse of a number  $x$  can be defined as the value  $x^{-1}$  such that  $x^{-1}x \equiv 1$ .

The following sections will discuss efficient algorithms for the implementation of addition, subtraction, multiplication and inversion in these fields. When implementing an elliptic curve system an important consideration is how to implement the underlying field arithmetic. Two questions of particular importance are whether to use even or odd characteristic fields and secondly whether to restrict implementation to fields of special types for efficiency or support any type of field.

**3.6 Fields of odd characteristic:** Field of odd characteristic is denoted with symbol  $F_p$  where  $p$  a large prime is. In this type, field elements will be represented as integers in the range  $0, 1 \dots p-1$  subjected to the arithmetic modulo  $p$ . Multiplication and division are challenging operations in  $F_p$  as compared to addition and subtraction. This section will discuss the techniques for performing fast modular reduction algorithms.

**3.6.1 Moduli of Special form or pre computed moduli [71]:** This algorithm is performed with the help of by shift and add operations and a multiplication by  $a$  removing any need of division. Modular inversion is the most cumbersome operation and is generally performed by using Euclidean algorithm. The modular inversion operation can be bypassed with the use of proper underlying curve arithmetic. Further improvement can be obtained with  $a$  having low Hamming weight i.e. few non zero elements in its binary representation.

**Algorithm 3. 7 Reduction Modulo**

---

**Algorithm 3.1      Reduction Modulo**

---

*ReductionModulo*  $p = b^t - a$ .  
*Input* : An integer  $x$ .  
*Output*:  $r = x \pmod{p}$ .  
1.  $q_0 \leftarrow \lfloor x / b^t \rfloor, r_0 \leftarrow x - q_0 b^t, r \leftarrow r_0, i \leftarrow 0$ .  
2. *while*  $q_i > 0$  *do*:  
3.  $q_{i+1} \leftarrow \lfloor q_i a / b^t \rfloor, r_{i+1} \leftarrow q_i a - q_{i+1} b^t$ ,  
4.  $i \leftarrow i + 1, r \leftarrow r + r_i$ .  
5. *while*  $r \geq p$  *do*  $r \leftarrow r - p$ .  
6. *Return*  $r$ .

---

The performance can also be improved by use of pre computed tables for performing the modular reduction. In this case every person who in the systems has to use curves defined over same finite field. This raises problem of interoperability.

### 3.6.2 Residue number system arithmetic

Residue Number System arithmetic (RNS) is very old concept based on the Chinese Remainder Theorem (CRT). For a modulus  $p$  a set of auxiliary primes  $p_i$  are chosen such that,

$$\prod_{i=1}^t p_i > p^2.$$

We then represent the element  $x$  modulo  $p$  as the vector  $(x_1, \dots, x_t)$  where,

$$x \equiv x_i \pmod{p_i}.$$

To add and multiply numbers we are only computing the addition and multiplication of their components of size very much smaller than the original modulus. The final result is obtained by the CRT.

### 3.6.3 Barrett Reduction [71]

When using Barrett reduction the standard multi precision methods are used for integer arithmetic operations. However the modular reduction is performed in a rather efficient way.

We assume that we are given a positive integer  $x$  which is of the size at most  $p^2$ . We wish to compute  $x \pmod{p}$ . As a pre computation we compute

$\mu = \lfloor b^{2t} / p \rfloor$  where  $b^t > p > b^{t-1}$  and  $b$  is the base size of the computer.

---

Algorithm 3.2 Barrett Reduction Algorithm

---

*Barrett Reduction*

*Input:*  $x, p$  and  $\mu$  such that  $x < b^{2t}, b^{t-1} < p < b^t$  and  $\mu = \lfloor b^{2t} / p \rfloor$

*Output:*  $z = x \pmod{p}$

1.  $q_0 \leftarrow \lfloor x / b^{k-1} \rfloor$

2.  $q \leftarrow \lfloor (\mu q_0) / b^{k+1} \rfloor$

3.  $r_1 \leftarrow x \pmod{b^{k+1}}, r_2 \leftarrow qp.$

4.  $z \leftarrow r_1 - r_2.$

5. *If*  $z < 0$  *then*  $z \leftarrow z + b^{k+1}.$

6. *While*  $z \geq m$  *do*  $z \leftarrow z - m.$

7. *Return*  $z.$

---

### 3.6.4 Montgomery Reduction [72]

This is the most successful method to implement arithmetic modulo a large prime  $p$ . Assume that  $b$  is the word base and  $t$  and  $R$  are defined by the equation,

$$R = b^t > p.$$

Every element  $x \in F_p$  is represented as  $xR \pmod{p}$  in *Montgomery representation*. Multiplication is much faster in this type of representation. The following algorithm explains the procedure for Montgomery Reduction. To compute  $x$  we first compute  $u$  which is easy once  $p^{-1} \pmod{R}$  has been computed, since  $R$  is a power of word base. Thus  $u$  can be computed with no modulo divisions, the computation of  $p^{-1} \pmod{R}$  being done once and stored for later use. To compute  $z$  we need to perform one further multi-precision multiplication, multi-precision addition and then a division by  $R$ . But division by  $R$  is a simple shift of words in  $y + up$  to the right by  $t$  spaces, since  $R = b^t > p$ .



---

**Algorithm 3.3 Montgomery Reduction Simple Case**

---

*Input* : A number  $y < pR$ .

*Output* :  $x = yR^{-1}(\bmod p)$ .

1.  $u \leftarrow -yp^{-1}(\bmod R)$ .

2.  $x \leftarrow (y + up) / R$ .

3. If  $x \geq p$  then  $x \leftarrow x - p$ .

4. Return  $x$ .

---

In fact we can even more efficient by setting  $p' = -p^{-1}(\bmod b)$  and if  $y$  is given by  $y = (y_{2t-1}, \dots, y_1, y_0)_b = y_{2t-1}b^{2t-1} + \dots + y_1b + y_0$  then  $yR^{-1}(\bmod p)$  can be computed by performing the following steps. The modified Montgomery reduction scheme is given below in which operation I step 2 can be performed using a single precision multiplication operation. The operation in step 3 is preformed by a shift, a scalar multiplication and then adds. Hence to get output we need to perform a set of multi precision addition, scalar multiplication and shift operation.

**Algorithm 3. 10 Modified Montgomery Reduction**

---

**Algorithm 3.4 Modified Montgomery Reduction**

---

*Input : A number  $y < pR$ .*

*Output :  $z = yR^{-1} \pmod{p}$ .*

*1. For  $i = 0$  to  $t - 1$  do :*

*2.  $u \leftarrow y_i p' \pmod{b}$ ,*

*3.  $y \leftarrow y + upb^i$ .*

*4.  $z \leftarrow y / R$ .*

*5. If  $z \geq p$  then  $z \leftarrow z - p$ .*

*6. Return  $z$ .*

---

To compute  $p' = -p^{-1} \pmod{b}$ , the extended Euclidean algorithm can be used. However it is rather easy to compute  $x^{-1} \pmod{2^w}$  using the following algorithm. To verify the correctness of this algorithm note that at the end of every execution step 3 we have  $xy \equiv 1 \pmod{2^i}$ . This method is very efficient as only single precision arithmetic is used assuming  $2^{w \leq b}$ .

**Algorithm 3. 11** *Computing  $x^{-1} \pmod{2^w}$*

---

**Algorithm 3.5** *Computing  $x^{-1} \pmod{2^w}$*

---

*Input : An odd int  $x, 0 < x < 2^w$ .*

*Output :  $y = x^{-1} \pmod{2^w}$ .*

*1.  $y \leftarrow 1$ .*

*2. For  $i = 2$  to  $w$  do :*

*3. If  $2^{i-1} < xy \pmod{2^i}$  then  $y \leftarrow y + 2^{i-1}$ .*

*4. Return  $y$ .*

---

Suppose two elements  $x, y \in F_p$  are given in the Montgomery representation i.e.  $X = xR \pmod{p}$ , and  $Y = yR \pmod{p}$ . To compute  $Z = xyR \pmod{p}$  first compute the standard multi precision multiplication of  $X$  and  $Y$  to obtain  $Z' = xyR^2$  which is a number of size at most  $p^2 < pR$ . By applying Montgomery reduction to the number  $Z'$  we can obtain

$Z$ . Thus to multiply two elements in Montgomery representation we need only perform a single multi precision multiplication followed by a Montgomery reduction. No divisions are needed. The operation can be made more efficient by interleaving the multiplication and reduction steps. Assume that  $X$  and  $Y$  are given in the form  $(x_{t-1}, \dots, x_0)_b$  and  $(y_{t-1}, \dots, y_0)_b$ .

To compute the Montgomery product  $Z = XYR^{-1} \pmod{p}$  perform the following-

---

**Algorithm 3. 12 Montgomery Multiplication**

---



---

**Algorithm 3.6 Montgomery Multiplication**

---

*Input :  $X$  and  $Y$*   
*Output :  $Z = XYR^{-1} \pmod{p}$ .*  
1.  $Z \leftarrow 0$ .  
2. *For  $i = 0$  to  $t - 1$  do :*  
3.  $u \leftarrow (z_0 + x_i y_0) p' \pmod{b}$ ,  
4.  $Z \leftarrow (Z + x_i y + up) / b$ .  
5. *If  $Z \geq p$  then  $Z \leftarrow Z - p$ .*  
6. *Return  $Z$ .*

---

The computation of  $u$  in above algorithm can be performed by is single precision and the computation of  $Z$  in step 4 requires two multiplications of multi precision integers by a word ,then two multi precision addition followed by right shift. Division in Montgomery representation can be performed by using the binary extended Euclidean algorithm and can be modified to compute to compute Montgomery inverse.

### 3.7 Fields of characteristic two

Finite fields of characteristic 2 are attractive to implement due to their ‘carry free’ arithmetic and the availability of different equivalent representations of the field, which can be adapted and optimized for the computational environment at the hand. This section will discuss arithmetic over the finite field  $F_{2^n}$   $n \geq 1$ . Field elements are represented as binary vectors of dimension  $n$  relative to the basis  $(\alpha_0, \alpha_1, \dots, \alpha_{n-1})$  of  $F_{2^n}$  as linear space over  $F_2$ . Field addition and subtraction are implemented as component wise exclusive OR (XOR), while implementation of multiplication and inversion depends on the basis chosen.

**3.7.1 Polynomial Bases:** A polynomial basis is one of the forms  $(1, \alpha, \alpha^2, \dots, \alpha^{n-1})$  where  $\alpha$  is a root of an irreducible polynomial  $f(x)$  of degree  $n$  over  $F_2$ . The field is then realized as  $F_2[x]/\langle f(x) \rangle$ , and the arithmetic is that of polynomial of degree at the most  $n-1$ , modulo  $f(x)$ .

**a. Modular reduction [57]:** By choosing  $f(x)$  as a low weight polynomial i.e. one with least possible number of non-zero coefficients, reduction modulo  $f(x)$  becomes very simple operation that is performed in time  $O(W_n)$  where  $W$  is the weight of the polynomial. For the purpose of practical interest it can be assumed that  $f(x)$  is either trinomial or pentanomial (i.e.  $W=3$  or  $5$ ). The following algorithm exemplifies reduction of a polynomial of degree  $2n-2$  such as is obtained from the product of two polynomials of degree  $n-1$ , modulo a trinomial  $f(x)$ .

### Algorithm 3. 13 Reduction Modulo

**Algorithm 3.7** *ReductionModulo*  $f(x) = x^n + x^t + 1, 0 < t < n$ .

---

*Input* :  $a(x) = a_0 + a_1x + a_2x^2 + \dots + a_{2n-2}x^{2n-2} \in F_2[x]$ .

*Output* :  $r(x) \equiv a(x) \pmod{f(x)}, \deg r(x) < n$ .

1. *For*  $i = 2n - 2$  *to*  $n - 1$  *do* :

2.  $a_{i-n} \leftarrow a_{i-n} + a_i, a_{i-n+t} \leftarrow a_{i-n+t} + a_i$ .

3. *Return*  $r(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$ .

---

The above algorithm operates on  $a(x)$  in place obviating need for extra storage for the results  $r(x)$ . Also in the software environment the algorithm is easily adapted to operate on computer words. If  $n - t \geq w$  where  $w$  is the word size, then algorithm scans the words containing the coefficients  $a_{2n-2}, a_{2n-1}, \dots, a_n$  from higher order to lower order, adding each word in to two positions offset  $n - t$  and  $n$  bits back respectively. The condition on  $n - t$  guarantees that a word does not add to any part of itself, and is thus processed only once. Each offset location requires up to two word XOR operations. The total number of word XOR operations in the trinomial case is therefore at the most  $4[n/w]$ . In general reduction modulo an irreducible of weight  $W$  requires at most  $2(W - 1)[n/w]$  word XOR operations. Another favored choice of irreducible polynomial is one of the form  $f(x) = x^n + g(x)$  Where the degree of  $g(x)$  is small relative to  $n$ .

**b. Multiplication:** When using polynomial bases, the first stage in computing the product of two elements of  $F_{2^n}$  is the multiplication of two polynomial of degree at most  $n-1$  in  $F_2[x]$ . This is carry free version of two  $n$ -bits integer. For multiplication the old and well tried recursive subdivision method first described for integers by Karatsuba [73] is often appropriate. Assume  $n$  is even. To compute the product  $a(x)b(x)$  where  $a(x), b(x) \in F_2[x]$  have degree  $n-1$ , we write

$$a(x)b(x) = (A_1(x)X + A_0(x)) (B_1(x)X + B_0(x)),$$

where  $A_0, A_1, B_0, B_1$  are polynomial of degree  $n/2-1$ , and  $X = x^{n/2}$ . The right hand side of the equation can be regarded as the product of two linear polynomials in  $X$ , with coefficients in  $F_2[x]$ . This product can be derived from the three products  $A_0B_0, A_1B_1$ , and  $(A_0 + A_1)(B_0 + B_1)$  of polynomials of degree  $n/2-1$ , i.e. one problem of size  $n$  is solved by solving three problems of size  $n/2$ . Similarly when  $n$  is odd, a problem of size  $n$  can be reduced to one of size  $(n-1)/2$  and two of size  $(n+1)/2$ . In either case proceedings recursively leads to an overall number of operations  $O(n^{\log_2 3})$ . As a final remark on multiplication in polynomial representation squaring is much easier than general multiplication in  $F_2[x]$ . To square a polynomial we just thin it out, inserting a zero between every two original binary coefficients.

**c. Inversion** In polynomial representations the extended Euclidean algorithm is always preferred for computing inverses. As with multiplication, fast symptotic algorithms are available for this computation. An  $O(M(n) \log n)$  algorithm is used for computing greatest common divisors, where  $M(n)$  denotes the complexity of multiplying  $n$ -bit polynomials. This algorithm can be easily modified to compute modular inverses. But, again, asymptotically fast methods becomes effective at very large values of  $n$ , usually beyond those used in WSN security. Therefore in WSN where values of  $n$  are moderate often rely on variants of the binary extended Euclidean algorithm for polynomials [74].

In any case,  $F_2^n$  inversion is often significantly slower than multiplication. In fact, an inversion can be easily replaced by multiplications chain. Such schemes derive from the field equation, which can be recast as  $\beta^{-1} = \beta 2^{n-2} = (\beta^{2^{n-1}} - 1)^2$  for all  $\beta \neq 0$  in  $F_2^n$ .

**3.7.2 Normal bases.** A normal basis of  $F_2^n$  over  $F_2$  has the form  $(\alpha, \alpha^2, \alpha^{2^2}, \dots, \alpha^{2^{n-1}})$  for some  $\alpha \in F_2^n$ . It is well known that such bases exist for all  $n \geq 1$ . Normal bases are useful hardware friendly. First, the field squaring problem is insignificant in normal basis representation, as squaring can be done by just a cyclic shift of the binary vector representing the input operand. A measure of the hardware complexity of such a multiplier is given by the number  $C_\alpha$  of ones in the  $n \times n$  binary matrix  $T = (T_{ij})$  defined by

$$\alpha^{1+2^i} = \sum_{j=0}^{n-1} T_{ij} \alpha^{2^j}, 0 \leq i \leq n-1.$$

The matrix  $T$  completely determines the structure of multiplication for the normal basis, as it captures all the information on products of basis elements. It is clear that  $C_\alpha \leq n^2$ . On the other hand,  $C_\alpha$  satisfies the lower bound  $C_n \leq 2n-1$ . When the lower bound is attained,  $\alpha$  is said to generate an optimal normal basis (ONB). An alternative characterization states that  $\alpha$  generates an ONB if and only if for all  $i_1, i_2, 0 \leq i_1, i_2 \leq n-1$ , there exist integers  $j_1, j_2$  such that  $\alpha^{2^{i_1}} + \alpha^{2^{i_2}} = \alpha^{2^{j_1}} + \alpha^{2^{j_2}}$ . It is easy to verify the definitions are equivalent.

The existence of *optimal normal bases*, an ONB of  $F_2^n$  over  $F_2$  exists if and only if one of the following conditions hold:

(i)  $n+1$  is prime, and 2 is primitive in  $F_{n+1}$ ; then the  $n$  non-trivial  $(n+1)$ st roots of unity form an ONB of  $F_2^n$  over  $F_2$ , called a Type I ONB;

(ii)  $2n+1$  is prime, and either

(1) 2 is primitive in  $F_{2n+1}$  or

(2)  $2n+1 \equiv 3 \pmod{4}$  and the multiplicative order of 2 in  $F_{2n+1}$  is  $n$ ; that is 2 generates the quadratic residues in  $F_{2n+1}$ ; then,  $\alpha = \gamma + \gamma^{-1}$  generates an ONB of  $F_2^n$  over  $F_2$ , where  $\gamma$  is a primitive  $(2n-1)$ st root of unity; this is called a Type II ONB.

The above characteristics show that ONB bases are having hardware complexity advantages.

An ONB is always self-dual, i.e.,  $\text{Tr}_{q|2}(\alpha_i \alpha_j) = 1$  if and only if  $i = j$ , where  $\alpha_i, \alpha_j$  denote arbitrary basis elements, and  $\text{Tr}_{q|2}(z)$  denotes the trace of  $z \in F_q$  over  $F_2$ ,  $q = 2$ . Also when an ONB of  $F_2^n$  exists, it is unique. Finally, the matrix  $T$  can be constructed directly from properties of the residue classes of integers modulo  $n+1$  (Type I) or  $2n+1$ . Thus, the field algebra can be realized directly without first requiring the construction of a binary irreducible polynomial of degree  $n$ . The bit-serial multipliers that are very effective for ONBs in hardware do not always map nicely to efficient software implementations, as single bit operations are expensive in the software. Also, while efficient bit-serial implementations are available for multiplication in ONB representation, they do not carry to inversion operations. It turns out, however, that by applying simple permutations, operations on ONB representations of both Types I and II can be handled through polynomial arithmetic, in a manner similar to the case of standard bases.

For Type I ONBs, we observe that the minimal polynomial of  $\alpha$  is  $f(x) = x^n + x^{n-1} + \dots + x + 1$ , and the set  $\{\alpha, \alpha^2, \alpha^{2^2}, \dots, \alpha^{2^{n-1}}\}$  is the same as the set  $\{\alpha, \alpha^2, \alpha^3, \dots, \alpha^n\}$ . Therefore, for an element with coordinate  $(a_0, a_1, \dots, a_{n-1})$  in ONB representation, we can write



$$\sum_{i=0}^{n-1} a_i \alpha^{2^i} = \sum_{j=1}^n a_{\pi(j)} \alpha^j,$$

Where the bijection  $\pi: \{1, 2, \dots, n\} \rightarrow \{0, 1, \dots, n-1\}$  is defined so that  $\pi(j) = i$  whenever  $2^i \equiv j \pmod{n}$ . Thus, after suitable permutation, we can operate on elements in ONB representation as polynomials modulo  $f(x)$  or even simpler, modulo  $(x+1)f(x) = x^{n+1} + 1$ . The latter will give result expressed in terms of  $1, \alpha, \alpha^2, \dots, \alpha^n$ , which are brought back to the desired basis set by using, when needed, the equality  $1 = \alpha + \alpha^2 + \dots + \alpha^n$ .

A similar, albeit slightly more involved transformation for TYPE II ONBs is described by Blake et al. Write  $P = 2n + 1$ , where  $n$  satisfies condition (II) above, and let  $\gamma$  be a  $P$ th root of unity. Let  $\Phi$  denote the vector space of all polynomials over  $F_2$  of the form  $a(x) = \sum_{j=1}^{2n} a_j x^j$ , where  $a_j = a_{p-j}$  for  $j = 1, 2, \dots, n$ . We call the elements of  $\Phi$  palindromic polynomials. In a palindromic representation of  $F_2^n$ , each field element corresponds to a palindromic polynomial. Addition is defined in the usual way, and the product of two palindromic polynomials  $a(x), b(x) \in \Phi$  is the unique polynomial  $C(x) \in \Phi$  such that

$$C(x) \equiv a(x) \cdot b(x) \pmod{x^P - 1}.$$

When we substitute  $x = \gamma$  in  $a(x)$ , we obtain

$$a(\gamma) = \sum_{j=1}^{2n} a_j \gamma^j = \sum_{j=1}^n a_j (\gamma^j + \gamma^{-j}).$$

It follows from condition (ii) that for every  $j \in \{1, 2, \dots, n\}$ , exactly one element in the pair  $\{j, p - j\}$  can be written as  $2^i$  modulo  $p$ , for some  $0 \leq i \leq n - 1$ . Hence, we can write

$$a(\gamma) = \sum_{i=0}^{n-1} a_{2^i} (\gamma^{2^i} + \gamma^{-2^i}) = \sum_{i=0}^{n-1} a_{2^i} (\gamma = \gamma^{-1})^{2^i} = \sum_{i=0}^{n-1} a_{2^i} \alpha^{2^i}.$$

where all indices are taken modulo  $p$ . The above equation implies that, up to permutation, the elements  $a_1, a_2, \dots, a_n$  are the coefficients of the ONB representation of  $a(\gamma)$ . It follows from this simple relationship between the coefficients of  $a(x)$  and the ONB representation of  $a(\gamma)$  that arithmetic operations in ONB representation can be realized as polynomial operations modulo  $x^p - 1$ . In particular, inverses in ONB representation can be computed using the Euclidean algorithm.

As an example of the transformation for Type II ONBs, consider the case  $n = 5$ . It is readily verified that the case satisfies Condition (ii), with 2 being primitive modulo  $p = 11$ . We have  $(2^0, 2^1, 2^2, 2^3, 2^4) \equiv (1, 2, 4, -3, 5) \pmod{11}$ . Thus an element  $a_0\alpha + a_1\alpha^2 + a_2\alpha^4 + a_3\alpha^8 + a_4\alpha^{16}$  in ONB representation corresponds to the palindromic polynomial

$$a_0x + a_1x^2 + a_3x^3 + a_2x^4 + a_4x^5 + a_4x^6 + a_2x^7 + a_3x^8 + a_1x^9 + a_0x^{10}.$$

**3.7.3 Subfield bases.** When  $n = n_1 n_2$ , we can regard  $F_{2^n}$  as an extension of degree  $n_2$  of  $F_{2^{n_1}}$ , and represent elements of  $F_{2^n}$  using a basis of the form  $\{a_1, a_2 \mid 0 \leq i < n_1, 0 \leq j < n_2\}$ ,

where  $\beta_0, \beta_1, \dots, \beta_{n_2-1}$  form a basis of  $F_{2^n}$  over  $F_{2^{n_1}}$  and  $\alpha_0, \alpha_1, \dots, \alpha_{n_1-1}$  form a basis of  $F_{2^{n_1}}$  over  $F_2$ . Thus, arithmetic can be done in two stages, with an ‘outer’ section doing operations on elements of  $F_{2^n}$  as vectors of symbols from

$F_{2^{n_1}}$ ; and an ‘inner’ section performing the operations on the symbols as binary words. Any combination of bases can be used, e.g., normal basis for the outer section, and polynomial basis for the inner one. The subfield representation is particularly advantageous when  $n_1$  is large enough so that  $n_2$  is small, but  $n_1$  is still small enough so that symbol operations can be made very fast in the computational environment at hand, e.g., by implementing the  $F_{2^{n_1}}$  arithmetic through look-up tables. Values of  $n_1$  between, say 4 and 16 are typical. The  $F_{2^n}$  inversion operation benefits the most from the structure, as the Euclidean algorithm is performed on much shorter polynomials, and the scheme benefits from the parallelization resulting from operations on symbols. Thus, typically, the gap between the running times of inversion and multiplication is smaller when a subfield representation is used, as compared to a polynomial basis over  $F_2$ . The latter, is a special case of subfield representation with  $n_1 = 1$ .

Inversion methods based on repeated multiplication can also be made more efficient when a subfield is available. Here, for any non-zero  $\beta \in F_{2^n}$ , we can write ,

$$\beta^{-1} = \frac{\beta^{s-1}}{\beta^s}, \quad \text{where } n = (2^n - 1) / (2^{n_1} - 1). \quad \text{The key observation}$$

is that  $\beta^s$  is in the subfield  $F_{2^{n_1}}$ . Hence, to compute  $\beta^{-1}$ , we obtain first  $\beta^{s-1}$  with an

optimized addition chain, and then  $\beta^s$  with an additional multiplication. The quotient in above equation is finally obtained with an inverse in  $F_{2^n_1}$  and a scalar multiplication by the resulting subfield element. Besides their advantages in implementing finite field arithmetic, subfields may help in two other central problems in elliptic curve cryptosystems: curves whose coefficients are in subfield allow for easier determination of the group order, and they offer ‘shortcuts’ for the important point multiplication.

### 3.8 Summary

## Chapter 4 Optimization of Coordinate System for ECC on WSN

---

### 4.1 Introduction

One of the crucial decisions when implementing an efficient ECC on WSN is deciding which coordinate system going to use. The coordinate system determines the efficiency of point addition and doubling algorithms on the elliptic curve, and hence the efficiency of the basic cryptographic operation of scalar multiplication [4].

The need for different coordinate systems have arisen due to the large amount of time required to complete a modular inversion operation. *Affine coordinate* system requires an inversion operation in both point addition and doubling. This operation is computationally very intense. Table 4.1 shows that even optimized assembly implementation in highly parallel processors require a significant number of cycles for modular inversion operation [75].

Table 4. 1 Comparison of binary and prime field on Motorola processor

| Finite Field Arithmetic Comparison in terms of Processor Cycles per field operation |                                 |                        |
|---|---------------------------------|------------------------|
| Field   | 163-Bits binary field           | 192-Bit prime field    |
| Operation   | $x^{163} + x^7 + x^6 + x^3 + 1$ | $x^{192} - x^{64} - 1$ |
| Inversion   | 7,752                           | 23,146                 |
| Multiplications   | 2,812                           | 330                    |
| Squaring  | 135                             | 213                    |

\*Cycle count for optimized assembly routines of Star core 140VLIW DSP processor (Motorola/Lucent 1999)

It provides results for the elliptic curve point operation using the elliptic curve  $y^2 + xy = x^3 + ax^2 + b$  over 163 bits and binary field using prime-polynomial  $x^{163} + x^7 + x^6 + x^3 + 1$ , a standard in cryptography. The second and third columns indicate the cycle times for each operation. This implementation uses optimized assembly programs for the start \*core SC140DSP processor, a highly parallel VLIW processor (Star\*Core 1999). The inversion operation is the slowest even in the parallel implementation. Recall that point addition or point doubling, in either binary or prime fields requires  $1I+2M+1S$  and  $1I+2M+2S$ , respectively where I, M and S refer to the number of Inversion, Multiplications, and squaring respectively.

For current standard with 163 bits point multiplication, one point double would require approximately 14,000 cycles and a point addition 13617 cycles on DSP processor. Additional cycles come from optimized field addition, modular reductions etc. which are generally insignificant compared to major operations listed namely multiplication, squaring and inversion.

To provide further illustration of the computational efforts in a 163 bits scalar multiplication over this curve ( computing  $kP$  where  $k$  is a 163 bit key), assuming half the bits are non zero, would requires 163 doubles and 81 additions totaling over three million cycles. At 350 MHz, this DSP processor can execute a typical 163 bits scalar multiplication in approximate 10 ms using *affine ordinate* system. It is interesting to note that the ratio of inversion cycles to field multiplication cycles (I/M) is 7:1 for prime field and 3:1 for binary field as shown in Table 4.1.

In general it is assumed that inversion operation takes seven fold longer times than a field multiplication, though in some platforms it may take 24 fold more execution time [75]. For example, prime field implementation on PC exhibits I/M ratios greater than 30.

The next section will illustrate how coordinate representation can optimize the performance, by eliminating the need for the inversion operation during point addition and point doubling.

## 4.2 Coordinate Systems in Elliptical Curve

An elliptic curve point  $P$ , can be represented with numerous coordinate systems. The prominent coordinate system used in ECC are *affine* coordinate, *projective* coordinate, *mixed* coordinate, *Jacobian* coordinate or *modified Jacobian* coordinate, in which the speed of point addition and point doubling differs. This objective of this chapter is to recommend efficient coordinate system for WSN platform so that standard cryptographic protocols like ECDH, ECDSA can be executed in shortest time. We have considered elliptical curves over prime fields except wherever mentioned binary. At the end of chapter we have suggested strategies for choosing optimal coordinate system for WSN depending on the software and hardware environment of these special networks.

### 4.2.1 Affine coordinates ( $A$ )

Affine coordinate system is the basic coordinate system used in the ECC [15, 57, 76]. Let us assume elliptical curve  $E$  with Weierstrass equation  $y^2 = x^3 + ax + b$  with  $a, b \in GF(p), p > 3$ . Let  $P_1 = (x_1, y_1)$  and  $P_2 = (x_2, y_2)$  be points in  $GF(p)$  given in *affine coordinates*, and where some convention is used to represent point at  $\theta$ . Assume that  $P_1, P_2 \neq \theta$  and  $P_1 \neq -P_2$ , the sum  $P_3 = (x_3, y_3) = P_1 + P_2$  can be computed as per below.

---

*Algorithm 4.1 Point Addition and Point Doubling in Affine Coordinate System*

---

**Point Addition**

$$\begin{aligned}\lambda &= \frac{y_2 - y_1}{x_2 - x_1}. \\ x_3 &= \lambda^2 - x_1 - x_2. \\ y_3 &= (x_1 - x_3)\lambda - x_3 - y_1.\end{aligned}$$

**Point Doubling**

$$\begin{aligned}\lambda &= \frac{3x_1^2 + a}{2y_1}. \\ x_3 &= \lambda^2 - 2x_1. \\ y_3 &= (x_1 - x_3)\lambda - x_3 - y_1.\end{aligned}$$

---

As seen from the above equations the cost of point doubling is  $1I + 4M$  and point addition is  $1I + 3M$ . We can ignore the cost of field additions as well as cost of multiplication by small constants. As seen from above equations, point addition and doubling requires *modular inversion* which is highly expensive as shown in Table 4.1. Modular Inversion can be avoided with the use of projective coordinate as discussed in the next section.

**4.2.2 Projective Coordinate  $P$**

*Projective coordinates* are also well known, although sometimes called *conventional projective coordinates* [15, 57, 76]. In projective coordinates the equation of  $E$  is,  $y^2z = x^3 + axz^2 + bz^3$ . The point  $(x_1 : y_1 : z_1)$  on  $E$  corresponds to the affine point  $(x_1/z_1, y_1/z_1)$  when  $z_1 \neq 0$  and to the point at  $\infty = (0:1:0)$  otherwise. The opposite of  $(x_1 : y_1 : z_1)$  is  $(x_1 : -y_1 : z_1)$ . Let  $P_1 = (x_1, y_1, z_1)$  and  $P_2 = (x_2, y_2, z_2)$  be points in  $GF(p)$  given



in projective coordinates, and where some convention is used to represent point at infinity  $\theta$ . Assume that  $P_1, P_2 \neq \theta$  and  $P_1 \neq -P_2$ , the sum  $P_3 = (x_3, y_3, z_3)$  can be computed as follow.

---

**Algorithm 4. 2 Point Addition and Point Doubling in Projective Coordinate**

---

*Algorithm 4.2 Point Addition and Point Doubling in Projective Coordinate*

---

**Point Addition**

$$\begin{aligned} \text{Set} \quad & A = y_2 z_1 - y_1 z_2, \\ & B = x_2 z_1 - x_1 z_2 \\ & C = A^2 z_1 z_2 - B^3 - 2B^2 x_1 z_2 \end{aligned}$$

$$\begin{aligned} \text{So That} \quad & x_3 = BC, \\ & y_3 = A(B^2 x_1 z_2 - C) - B^3 y_1 z_2 \\ & z_3 = B^3 z_1 z_2 \end{aligned}$$

**Point Doubling**

$$\begin{aligned} \text{Set} \quad & A = az_1^2 + 3x_1^2, \\ & B = y_1 z_1, \\ & C = x_1 y_1 B, \\ & D = A^2 - 8C \end{aligned}$$

$$\begin{aligned}
& x_3 = 2BD, \\
\text{So That } & y_3 = A(4C - D) - 8y_1^2 B^2, \\
& z_3 = 8B^3
\end{aligned}$$


---

The key observation from the above formulae is that point addition can be done using field multiplication only, with no inversion required. Thus inversions are deferred, and only one needed to be performed at the end of point multiplication operation, if it is required the final results be given in affine coordinates. The cost of eliminating inversions is an increased number of multiplications, so the appropriateness of using *projective coordinate* is strongly determined by the ratio I: M. As seen from the above formulae the cost of point addition is 14M and cost of point doubling is 12M. If one of the input points to addition is given by  $(x_2, y_2, 1)$  that is directly transformed from *affine coordinate* system, then requirement of addition decreases to 11M. In spite of faster execution time, the *projective coordinate* representation requires larger code size and more memory for storing pre computed values as compared to the affine coordinate system.

#### 4.2.3 Jacobian and Chudnovsky Jacobian Coordinates $(J, J^c)$

In Jacobian coordinates [18, 67] the equation of E is,  $y^2 = x^3 + axz^4 + bz^6$ . The point  $(x_1 : y_1 : z_1)$  on E corresponds to the affine point  $(x_1 / z_1^2, y_1 / z_1^3)$  when  $z_1 \neq 0$  and to the point at  $\infty = (1 : 1 : 0)$  otherwise. The opposite of  $(x_1 : y_1 : z_1)$  is  $(x_1 : -y_1 : z_1)$ . Let  $P_1 = (x_1, y_1, z_1)$  and  $P_2 = (x_2, y_2, z_2)$  be points in  $GF(p)$  given in projective coordinates, and where some convention is used to represent point at  $\theta$ . Assume that  $P_1, P_2 \neq \theta$  and  $P_1 \neq -P_2$ , The sum  $P_3 = (x_3, y_3, z_3)$  can be computed as follows.

---

*Algorithm 4.3 Point Addition and Point Doubling in Jacobian Coordinate*

---

**Point Addition**

Set  $A = x_1 z_2^2, B = x_2 z_1^2, C = y_1 z_2^3, D = y_2 z_1^3,$   
 $E = B - A, F = D - C.$

So That  $x_3 = -E^3 - 2AE^2 + F^2,$   
 $y_3 = -CE^3 + F(AE^2 - X_3),$   
 $z_3 = z_1 z_2 E$

**Point Doubling**

Set  $A = 4x_1 y_1^2,$   
 $B = 3x_1^2 + az_1^4$

So That  $x_3 = -2A + B^2,$   
 $y_3 = -8y_1^4 + B(A - x_3)$   
 $z_3 = 2y_1 z_1$

---

The cost of point addition in this coordinate system is  $12M + 4S$  and point doubling will cost  $4M + 6S$ . If one of the point is given as  $P_1 = (x_1, y_1, 1)$ , the cost of addition reduces to  $8M + 3S$ . If the value of  $a = -3$ , point doubling requires only  $4M + 4S$  operations. The parameter

$a = 0$  is more advantageous as the this cost further drops down to  $3M + 4S$ . However this choice is far more special and the endomorphism ring  $\text{End}(E)$  contains a third root of unity. In *Jacobian Coordinates*, doublings are faster and additions are slower than the *projective coordinates*. To improve additions, a point  $P$  can be represented as a quintuple  $(x_1, y_1, z_1, z_1^2, z_1^3)$ . These coordinates are called *Chudnovsky Jacobian Coordinates*. Additions and doublings are given by same formulae as above but the cost of addition is  $11M + 3S$  and for doubling  $5M + 6S$ .

#### 4.2.4 Modified Jacobian Coordinates ( $J^m$ )

*Modified Jacobian coordinates* were introduced by [77]. They are based on *Jacobian coordinates* but the internal representation of a point  $P$  is the quadruple  $(x_1, y_1, z_1, az_1^4)$ . The main difference in the formulae is  $C = 8y_1^4, y_3 = B(A - x_3) - C, az_3^4 = 2C(az_1^4)$ . An addition takes  $13M + 6S$  and a doubling  $4M + 4S$ . If one point is *affine coordinates*, an addition takes  $9M + 5S$ . This system offers fastest doubling procedure.

#### 4.2.5 Mixed Coordinates

The idea of *mixed coordinate* was proposed by [24] where the inputs and outputs to point addition and doublings may be in different coordinates. Mixed coordinates can be very efficient when scalar multiplication is implemented with the base point stored in affine coordinates [4]. This idea was already mentioned in adding an *affine point* to one in other system. In general, one can add points expressed in two different systems and give results in third one [77]. For example  $J + J^c = J^m$  means that we can add points in *Jacobian and Chudnovsky Jacobian coordinates* and express the results in *Modified Jacobian System*. In order to use mixed coordinate it is sometime necessary to convert a point representation from one coordinate system to another to have the input in the required format for the addition and doubling algorithm [4]. A point in *affine coordinate*  $(x, y)$  can be converted to any other coordinate by using the following equations. None of the above conversions requires modular squaring, multiplications or inversions making them quite fast.

**Algorithm 4. 4 Conversion of Affine coordinates to other coordinates**

---

*Algorithm 4.4 Conversion of Affine coordinates  $(x, y)$  to other coordinates*

---

|                 |               |                                       |
|-----------------|---------------|---------------------------------------|
| Affine $(x, y)$ | $\rightarrow$ | Projective $(x, y, 1)$                |
| Affine $(x, y)$ | $\rightarrow$ | Jacobian $(x, y, 1)$                  |
| Affine $(x, y)$ | $\rightarrow$ | Chudnovsky Jacobian $(x, y, 1, 1, 1)$ |
| Affine $(x, y)$ | $\rightarrow$ | Modified Jacobian $(x, y, 1, a)$      |

---

On the other hand conversion from a *projective coordinate* to any other coordinate is more expensive, requiring several multiplications or squaring in each case as shown in the following algorithm 4.5.

**Algorithm 4. 5 Conversion of projective coordinate to other coordinates[4]**

---

*Algorithm 4.5 Conversion of projective coordinate  $(x, y, z)$  to other coordinates*

---

|                        |               |   |
|------------------------|---------------|---|
| Projective $(x, y, z)$ | $\rightarrow$ | Affine $(xz^{-1}, yz^{-1})$                   |
| Projective $(x, y, z)$ | $\rightarrow$ | Jacobian $(xz, yz^2, z)$                      |
| Projective $(x, y, z)$ | $\rightarrow$ | Chudnovsky Jacobian $(xz, yz^2, z, z^2, z^3)$ |
| Projective $(x, y, z)$ | $\rightarrow$ | Modified Jacobian $(xz, yz^2, z, az^4)$       |

---

Conversion from a point in *Jacobian coordinate* to any of the other coordinate is generally more efficient than the *projective* case as shown below-

**Algorithm 4. 6 Conversion of Jacobian coordinates to other coordinates[4]**

---

*Algorithm 4.6 Conversion of Jacobian coordinates  $(x, y, z)$  to other coordinates*

---

|                      |   |   |
|----------------------|---|---|
| Jacobian $(x, y, z)$ | → | Affine $(xz^{-2}, yz^{-3})$               |
| Jacobian $(x, y, z)$ | → | Projective $(xz, y, z^3)$                 |
| Jacobian $(x, y, z)$ | → | Chudnovsky Jacobian $(x, y, z, z^2, z^3)$ |
| Jacobian $(x, y, z)$ | → | Modified Jacobian $(x, y, z, az^4)$       |

---

The equations for conversion from *Chudnovsky Jacobian* and *modified Jacobian* coordinates to any of the other coordinate system are the same as those for Jacobian coordinates. The number of operations required for each of these conversions between coordinate systems is shown in the following table 4.2.

**Table 4. 2 Cost of Conversion To and From Various Coordinate Systems [4]**

| <i>From /To</i>   | <i>Affine</i> | <i>Projective</i> | <i>Jacobian</i> | <i>Chudnovsky</i> | <i>Modified</i> |
|-------------------|---------------|-------------------|-----------------|-------------------|-----------------|
| <i>Affine</i>     | -             | -                 | -               | -                 | -               |
| <i>Projective</i> | 2M+I          | -                 | 2M+S            | 3M+S              | 3M+2S           |
| <i>Jacobian</i>   | 3M+S+I        | 2M+S              | -               | M+S               | 1M+2S           |
| <i>Chudnovsky</i> | 3M+S+I        | 1M                | -               | -                 | 1M+1S           |
| <i>Modified</i>   | 3M+S+I        | 2M+S              | -               | M+S               | -               |

\*  $M$ =Multiplication,  $I$ =Inversion,  $S$ =squaring

From the above table it is clear that conversion from *affine coordinate* is very efficient because the conversion only consists of setting all of the  $z, z^2, z^3$  coordinates to 1 and  $az^4$  coordinates to  $a$ .

On the other hand conversion to *affine coordinate* from any other coordinate system is inefficient because of the inversion involved.

Conversion to or from *projective coordinates* is mostly less efficient than converting between the *Jacobian variant* coordinates systems.

Conversion among the three *Jacobian variants* are the most efficient, and this thesis recommends use of them to provide the fastest *mixed coordinates* scalar multiplication algorithm for WSN platform.

The table 4.3 adopted from [24] gives cost of point doublings and point addition for various *mixed coordinate* system. In this table notation  $C_1 + C_2 = C_3$  is used. The points to be added are given as  $C_1, C_2$  and their sum is  $C_3$ . For e.g.  $A + A = J^m$  means, both the inputs are in *affine coordinate* and the output is in *modified Jacobian coordinate*. We can notice from the Table 4.3 that *Jacobian coordinates* yield the fastest point doubling, while *mixed Jacobian – affine coordinates* yield the fastest point addition [24]. As the table itself is self explanatory, this research does not feel any need for any practical implementations on WSN platform for verification of point doubling and point addition cost in various mixed coordinate system.

Table 4. 3 Cost of Operations In Mixed Coordinates [24]

| <i>Point Doubling</i> |             | <i>Point Addition</i> |             |
|-----------------------|-------------|-----------------------|-------------|
| <i>Operation</i>      | <i>Cost</i> | <i>Operation</i>      | <i>Cost</i> |
| $2P$                  | 7M + 5S     | $J^m + J^m$           | 13M+6S      |
| $2J^c$                | 5M + 6S     | $J^m + J^c = J^m$     | 12M+5S      |
| $2J$                  | 4M + 6S     | $J + J^c = J^m$       | 12M+5S      |
| $2J^m = J^c$          | 4M + 5S     | $J + J$               | 12M+4S      |
| $2J^m$                | 4M + 4S     | $P + P$               | 12M+2S      |
| $2A = J^c$            | 3M + 5S     | $J^c + J^c = J^m$     | 11M+4S      |
| $2J^m = J$            | 3M + 4S     | $J^c + J^c$           | 11M+3S      |
| $2A = J^m$            | 3M + 4S     | $J^c + J = J$         | 11M+3S      |
| $2A = J$              | 2M + 4S     | $J^c + J^c = J$       | 10M+2S      |
| -                     | -           | $J + A = J^m$         | 9M+5S       |
| -                     | -           | $J^m + A = J^m$       | 9M+5S       |
| -                     | -           | $J^c + A = J^m$       | 8M+4S       |
| -                     | -           | $J^c + A = J^c$       | 8M+3S       |
| -                     | -           | $J + A = J$           | 8M+3S       |
| -                     | -           | $J^m + A = J$         | 8M+3S       |
| -                     | -           | $A + A = J^m$         | 5M+4S       |
| -                     | -           | $A + A = J^c$         | 5M+3S       |
| $2A$                  | I + 2M + 2S | $A + A$               | I+2M+S      |



---

Example: Elliptical point in various coordinate systems

---

Let us assume  $E : y^2 = x^3 + 1132x + 278$ .

Let P (1120, 1391) and Q (894, 1425) are two affine points on E. These points in various coordinate systems can be represented as Table 4.4.

Table 4. 4 Representation of Point in Various Coordinate Systems[24]

| <i>System</i>        | <i>Equation</i>                  | <i>P</i>                 | <i>Q</i>                |
|----------------------|----------------------------------|--------------------------|-------------------------|
| <i>A</i>             | $y^2 = x^3 + 1132x + 278$        | (1120,1391)              | (894,1425)              |
| <i>p</i>             | $y^2z = x^3 + 1132xz^2 + 278z^3$ | (450 : 541 : 1449)       | (1774 : 986 : 1530)     |
| <i>J</i>             | $y^2 = x^3 + 1132xz^4 + 278z^6$  | (1213 : 408: 601)        | (1623 :504: 1559)       |
| <i>J<sup>C</sup></i> | -                                | (1213, 408, 601,661,667) | (1623,504,1559,842,713) |
| <i>J<sup>M</sup></i> | -                                | (1213,408,601,1794)      | (1623,504,1559,1232)    |

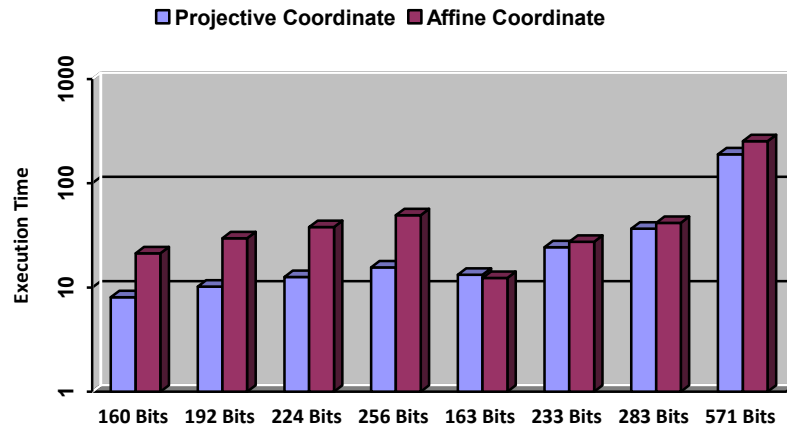
### 4.3 Implementation on MIRACL crypto Library

As a matter of further investigation, this research implemented ECDH and ECDSA protocols on MIRCAL crypto library only to show that modular inversion operation is expensive and must be avoided to accelerate the scalar multiplication process while choosing coordinate systems. ECDH and ECDSA protocols are important for authentication of nodes and pair wise key establishments in WSN security. Two coordinate system namely *affine and projective* were used to investigate effect of modular inversion operation on the execution time of these protocols. The experiments were performed on Intel Pentium IV processor working on 1.8 GHz of frequency and 768 MB of RAM. NIST recommended curves were

referred for the same with prime and binary field for various key sizes. It has been found that due to absence of modular inversion operation for point doubling and addition operation, *projective coordinates* offers better results than *affine coordinate* system. The results are given in Table 4.5. The graphical comparison is shown in Figure 4.1. The actual output windows are attached as an Appendix B with this thesis. The y-axis of Figure 4.1 uses logarithmic scale.

**Table 4. 5 Comparison of Projective and Affine Coordinate for ECDH and ECDSA Protocol for WSN**

| <i>Sr. No</i> | <i>Parameter</i>   | <i>Projective Coordinates (ms)</i> | <i>Affine Coordinates (ms)</i> |
|---------------|--------------------|------------------------------------|--------------------------------|
| 1             | 160 Bits $GF(p)$   | 8.06                               | 21.23                          |
| 2             | 192 Bits $GF(p)$   | 10.22                              | 29.72                          |
| 3             | 224 Bits $GF(p)$   | 12.56                              | 38.02                          |
| 4             | 256 Bits $GF(p)$   | 15.55                              | 49.33                          |
| 5             | 163 Bits $GF(2^m)$ | 13.25                              | 12.30                          |
| 6             | 233 Bits $GF(2^m)$ | 24.27                              | 27.40                          |
| 7             | 283 Bits $GF(2^m)$ | 36.81                              | 41.73                          |
| 8             | 571 Bits $GF(2^m)$ | 189.26                             | 253.50                         |



**Figure 4. 1 Evaluation of Coordinate System for ECDH and ECDSA protocol on MIRACL Crypto Library**

## 4.4 Summary

There are many choices of coordinate's representations and these have a significant impact on the computational cost of cryptographic protocols like ECDH, ECDSA. Table 4.3 shows the cost of computations for various mixed coordinates.

Based on Table 4.3 and various results obtained on MIRACL, this thesis recommends *AJM* coordinate system with one input in *affine coordinate* and other in *Jacobian coordinate* with output in *Modified Jacobian coordinate* for point addition algorithm. This coordinate system will offer best results if the constant  $a = -3$  chosen. (All curves used in MIRACL are having value of  $a = -3$  for this reason. The list of the NIST curves is given in the Appendix A of the thesis). One of the input is chosen in *affine coordinate* because of the faster implementations available and because fewer variables were required in this case.

For point doubling algorithm, this thesis recommends *Modified Jacobian MM* or *Mixed Jacobian MJ* coordinate system. Experiments carried out on MIRACL crypto library clearly indicate that modular inversion is the costly operation for WSN and it has to be avoided in any case. *AJM*, *MM* and *MJ* coordinate system do not use any modular inversion operation. Also conversions among three *Jacobian variants* are most efficient. All these circumstances leads to better performance on WSN node while implementing the ECDH and ECDSA protocols.

Other method for attempting to speed up the EC computations on WSN platform is to transform the base point of curve from *affine coordinate* to *projective coordinate*. The use of *projective coordinate* will eliminate the use of modular inversion operation. But in this case, one modular inversion will be needed at the end to convert final result back to *affine coordinate* system. (Please refer Table 4.2)

Although, mixed coordinates,  $A + A = J^m$ ,  $A + A = J^e$  looks very fast, these methods are not very useful for WSN scalar multiplication process. Because scalar multiplication consists of repeated additions and doublings and the output of an addition or doublings is never in *affine*

*coordinates* for an efficient algorithm. In order to use  $A + A = J^m$  and  $A + A = J^c$  operations in scalar multiplication, the input point must be converted to *affine coordinates*. This requires inversion and makes scalar multiplication algorithm computationally intense.

Doing *two in one scalar multiplication* ( $k_1P + k_2Q$ ) process which is always referred in literature as Shamir's trick [25] will improve the signature verification time for ECDSA algorithm. The results obtained for Shamir's trick on MIRACL library are given in the Appendix B of the thesis for verification of signature process. The Shamir's trick has been discussed thoroughly in Chapter 6.

The performance of  $AJM$ ,  $MM$  and  $MJ$  coordinate system can also enhanced by use of signed digit (SD) recoding method. This thesis proposes One's complement Subtraction (OCS) Method for signed digit recoding of scalar and is discussed at length in the next chapter 5.

## **Chapter 5 One's Complement Subtraction (OCS) algorithm proposed for Recoding of Integer on WSN Platform**

---

### **5.1 Introduction**

The ECC offers various advantages as compared to other conventional PKC techniques such as RSA [6], DSA ,DH[22] and so far there is a lack of sub exponential attack on ECC. The best known algorithm for solving ECDLP , takes the fully exponential time and for solving factorial algorithms like RSA and DH takes sub exponential time [19]. ECC offers the same level of security with smaller key size and it also leads to the better performance in limited environments like mobile phones, PDA, WSN [78]. The standard bodies such as IEEE, NIST, IETF and ISO has already endorsed ECC as an alternative and efficient public key cryptosystem.

While implementing ECC on WSN, the main operations such as key agreement, signature generations, signing and verifications involve scalar multiplication. The optimization of scalar multiplication process is very necessary as WSN consists of thousands of nodes communicating with each other with limited resources. Compared to traditional networks, WSN has many resource constraints. The MICA2 node [55] which is the basic unit of sensor network consists of an 8 bit microcontroller [79] working on 7.3 MHz frequency. As a result MICA2 nodes have limited computational power. Normally radio transceiver of MICA motes can achieve maximum data rate of 250 Kbits/sec which puts a limitation on the communication resources. The flash memory which is available on the MICA mote is only

512 Kbyte. Apart from these the battery which is available on the board is of 3.3.V with 2A-Hr capacity. Due to the above limitations scalar multiplication takes long time to execute on the wireless sensor node.

This chapter proposes a novel algorithm called ‘one’s complement subtraction’ (OCS) for recoding of integer in scalar multiplication process which will improve the performance of ECC on wireless sensor networks. Next subsection gives Elliptical Curve Diffie Hellman (ECDH) protocol [22] which makes use of scalar multiplication operation to calculate the public keys.

## 5.2 ECDH Protocol Implemented on WSN to Achieve Data Confidentiality

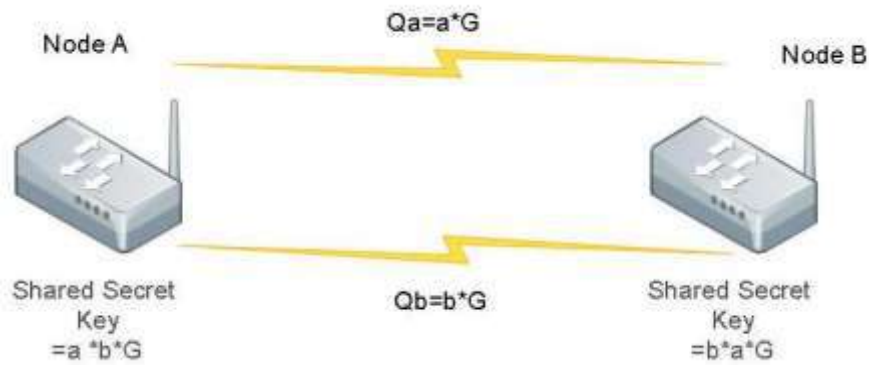


Figure 5. 1 ECDH Protocol Implemented on WSN

As shown in figure 5.1, Node A and Node B agrees on a particular curve  $E$  with base point  $G$ . They generate their public keys  $Q_a, Q_b$  by multiplying  $G$  with their private keys namely  $a$  and  $b$ . After sharing public keys they generate the shared secret key by multiplying public keys by their private keys. The shared secret key calculated by Node A and Node B are  $a * b * G$  and  $b * a * G$  respectively. With the known values of  $Q_a, Q_b$  and  $G$ , it is computationally intractable for an eavesdropper to calculate  $a$  and  $b$  which are the private keys of Node A and Node B. As a result, adversaries cannot figure out the shared secret key.

In this protocol the operation of multiplying  $G$  by integers  $a$  and  $b$  is called as scalar multiplication and is the most time consuming operation while implementing ECC on WSN networks. Gura et.al [13] showed that 85% of execution time is spent on scalar multiplication operation. The scalar multiplication is given by the formula,

$$Q = [a]G = \underbrace{G + G + \dots + G}_{a \text{ times}} \quad \text{Equation 5.1}$$

where  $G$  is a base point, and  $a$  is positive integer in the range  $1 \leq a < \text{ord}(P)$ . As shown in the equation V.I, scalar multiplication is done by successive point doubling and point addition operations. For some of the cryptographic protocols,  $G$  is a fixed point that generates a large prime order subgroup of  $E(F_q)$ , while for others  $G$  is dynamic point in such a subgroup. The strength of the cryptosystem lies in the fact that given the curve, the point  $G$  and  $[a]G$ , it is hard to recover  $a$ . This is called the Elliptic Curve Discrete Logarithm Problem (ECDLP) and the computation  $[a]G$  referred as scalar *multiplication*.

Operational efficiency of *scalar multiplications* is depends on the Hamming weight of  $a$  which is recoded in the form of '0' and '1'. The following section will take overview of existing recoding algorithms of scalar  $a$ . The number of point doubling and point additions in scalar multiplication depends on the recoding of integer  $a$ . Expressing integer  $a$  in binary format highlight this dependency. The number of '0' and number of '1' in the binary form, their places and the total number of bit affects computational cost of scalar multiplications. The Hamming weight i.e. the number of non zero elements, determines the number of point additions and bit length of integer  $a$  determines the number of point doublings operations in scalar multiplication.

## 5.3 The Existing Methods of Integer Recoding

### 5.3.1 Binary Method

This is the oldest method [80] in which the integer  $k$  is represented in binary form as

$$k = k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + \dots + k_1 + k_0$$

where  $k_{n-1} = 1$  and

$$k_i \in \{0,1\},$$

$$n = 0,1,2,3,\dots,n-1.$$

That is  $k = \sum_{j=0}^{\ell-1} k_j 2^j, k_j \in \{0,1\}$ . The Binary method [81] scans the bits of  $k$  either from left-to-right or right-to-left. The binary method for the computation of scalar multiplication is given in the Algorithm 5.1 and 5.2. Algorithm 5.1 processes bits of  $k$  from left to right and Algorithm 5.2 processes bits from right to left.

#### Algorithm 5.1 Left to Right Binary Method

---

#### Algorithm 5.1 Left to Right Binary Method

---

*Input* : Point  $P \in E(F_q)$ , an  $\ell$  bit integer  $k = \sum_{j=0}^{\ell-1} k_j 2^j, k_j \in \{0,1\}$

*Output* :  $Q = kP$

1.  $Q \leftarrow \infty$ . For  $j = \ell - 1$  to 0 do,

1.1  $Q \leftarrow 2Q$ ,

1.2 If  $k_j = 1$ , Then  $Q \leftarrow Q + P$

2. Return  $Q$ .

---

#### Algorithm 5.2 Right to Left Binary Method

---



---

#### Algorithm 5.2 Right to Left Binary Method

---



*Input* : Point  $P \in E(F_q)$ , an  $\ell$  bit integer  $k = \sum_{j=0}^{\ell-1} k_j 2^j, k_j \in \{0,1\}$

*Output* :  $Q = kP$

1.  $Q \leftarrow \infty$ . For  $j = 0$  to  $\ell - 1$  do,

1.1 If  $k_j = 1$ , Then  $Q \leftarrow Q + P$

1.2  $P \leftarrow 2P$ ,

2. Return  $Q$ .

---

The cost of multiplication in binary method depends on the number of non zero elements and length of the binary representation of  $k$ . If the MSB of binary representation is 1, then this method requires  $(l-1)$  point doublings and  $(w-1)$  where  $l$  is the length of the binary expansion and  $w$  is the Hamming weight of the  $k$  that is the number of non zero elements in expansion. For example if  $k = 1788 = (1101111100)_2$ , it will require  $w - 1 = 8 - 1 = 7$  point additions and  $l - 1 = 11 - 1 = 10$  point doublings operations.

The expected number of non zero elements in the binary representations of  $k$  is  $l/2$ , therefore expected running time of algorithm 5.1 is approximately  $l/2$  point additions and  $l$  point doublings denoted by,  $l/2A + lD$  where  $A, D$  indicates point addition and doublings respectively.

Let  $M, S, I$  denotes field multiplication, field squaring and field inversion respectively. If Affine coordinates are used, then the running time expressed in terms of field operation is

$2.5l S + 3l M + 1.5l I$  if the field  $E(F_q)$  is greater than 3 and,

$3/M + 1.5/I$  if the field  $E(F_q)$  is binary.

Whenever the bit is 1, two elliptical curve operations namely point addition and point doubling will be required and when the bit is 0, only point doubling operation is required. So if recoding method reduces number of 1 in integer recoding, it will reduce the number of addition operations and will speed up scalar multiplication process.

### 5.5.2 Signed Digit Representation Recoding

A.D.Booth [82] proposed in 1951 a new scalar representation called *signed binary method* and Rietweisner [83] proved that every integer can be represented in this way. The next section gives more details about this NAF recoding method.

### 5.5.3 Non Adjacent Form ( NAF) method

If  $P = (x, y) \in E(F_q)$  then  $-P = (x, x + y)$  if  $F_q$  is a binary field, and  $-P = (x, -y)$  if  $F_q$  has characteristic greater than 3. Thus subtraction of points on an elliptical curve is just as efficient as addition. This facilitates using a signed digit representations in which integer  $k = \sum_{i=0}^{l-1} k_i 2^i$ , where  $k_i \in \{0, \pm 1\}$ . When signed digit representation has no adjacent non zero digits, i.e. it is called non-adjacent form (NAF).

**Definition:** A non adjacent form (NAF) of a positive integer  $k$  is an expression  $k = \sum_{i=0}^{l-1} k_i 2^i$ , where  $k_i \in \{0, \pm 1\}$ ,  $k_{l-1} \neq 0$ , and no two consecutive digits  $k_i$  are non zero. The length of the NAF is  $l$ .

**Theorem:** The properties of NAF for integer  $k$  with the length  $l$  may be summarized as per below-

1.  $k$  has unique NAF denoted by  $\text{NAF}(k)$ .
2.  $\text{NAF}(k)$  has the fewest non zero digits of any signed digit representations of  $k$ .

3. The length of the  $\text{NAF}(k)$  is at most one more than the length of the binary representation of  $k$ .

4. If the length of  $\text{NAF}(k)$  is  $l$ , then  $2^l/3 < k < 2^{l+1}/3$ .

5. The average density of nonzero digits among all NAFs of length  $l$  is approximately  $1/3$ .

$\text{NAF}(k)$  can be efficiently computed using following Algorithm 5.2. The digits of  $\text{NAF}(k)$  are generated by repeatedly dividing  $k$  by 2, allowing remainders of 0 or  $\pm 1$ . If  $k$  is odd, then the remainder  $r \in \{-1, 1\}$  is chosen so that the quotient  $(k-r)/2$  is even –this ensures that the next NAF digit is 0.

---

**Algorithm 5.3** Computing the NAF of a positive integer

---



---

**Algorithm 5.2** Computing the NAF of a positive integer

---

Input: A positive integer  $k$ .

Output:  $\text{NAF}(k)$ .

1.  $i \leftarrow 0$ .
  2. *While*  $k \geq 1$  *do*
    - 2.1 *If*  $k$  is odd *then* :  $k_i \leftarrow 2 - (k \bmod 4), k \leftarrow k - k_i$ ;
    - 2.2 *Else* :  $k_i \leftarrow 0$ .
    - 2.3  $k \leftarrow k/2, i \leftarrow i + 1$ .
  3. *Return*  $(k_{i-1}, k_{i-2}, \dots, k_1, k_0)$ .
- 

Algorithm 5.3 modifies the left to right binary method for point multiplication by using  $\text{NAF}(k)$  instead of the binary representation of  $k$  and is called as *addition-subtraction*

method. The expected running time of Algorithm 5.3 is approximately  $l/3A + lD$  where  $A, D$  indicates point addition and doubling respectively. Point addition and point subtraction requires the same timings and therefore shown as only point addition in estimating time. NAF method reduces the Hamming weight of integer from  $l/2$  to  $l/3$ .

**Algorithm 5.4 Binary NAF method for point multiplication**

---

Algorithm 5.3 Binary NAF Method for point multiplication

---

*Input* : Point  $P \in E(F_q)$ , an  $\ell$  bit integer  $k = \sum_{i=0}^{\ell-1} k_i 2^i, k_i \in \{0, \pm 1\}$

*Output* :  $Q = kP$ .

1.  $Q \leftarrow \infty$ . For  $i = \ell - 1$  to 0 do,
    - 1.1  $Q \leftarrow 2Q$ ,
    - 1.2 If  $k_i = 1$ , Then  $Q \leftarrow Q + P$ .
    - 1.3 If  $k_i = -1$ , Then  $Q \leftarrow Q - P$ .
  2. Return  $Q$ .
- 

The following Algorithm 5.2 computes the converts binary number to NAF.

**Algorithm 5.5 Conversion from Binary to NAF**

---

Algorithm 5.2 Conversion from Binary to NAF

---

*Input* : Integer  $k = \sum_{j=0}^{\ell-1} k_j 2^j, k_j \in \{0, 1\}$ .

*Output* : NAF  $k = \sum_{j=0}^{\ell} s_j 2^j, s_j \in \{1, 0, -1\}$ .

1.  $c_0 \leftarrow 0$ .
  2. For  $j = 0$  to  $\ell$  do :
    3.  $c_{j+1} \leftarrow \lfloor (k_j + k_{j+1} + c_j) / 2 \rfloor$  (assume  $k_i = 0$  for  $i \geq \ell$ ),
    4.  $s_j \leftarrow k_j + c_j - 2c_{j+1}$ .
  5. Return  $(s_\ell s_{\ell-1} \dots s_0)$ .
-

NAF has usually fewer non zero digits than binary representations. The average hamming weight for NAF form is  $(n-1)/3$  [84, 85]. So generally it requires  $(n-1)$  point doublings and  $(n-1)/3$  point additions.

#### 5.3.4 Mutual Opposite Form (MOF) Method

This method was proposed by Okeya [86] which scans bits from left to right and eliminates the need of storing the multiplier in advance. In MOF recoding signs of adjacent non zero bits are opposite. The MSB and LSB non zero bits are 1 and -1 respectively. Any positive integer can be represented by unique MOF.

The  $n$ -bit binary string  $k$  is converted in to a signed binary string by computing  $mk = 2k - k$  where minus sign indicated bitwise subtraction. The conversion of MOF recoding of an integer is highly flexible because the conversion can be made either from right to left or left to right. The output of MOF is same as output of NAF. Algorithm 5.6 explains processing from binary to MOF recoding method.

---

Algorithm 5.6 Left to Right processing from Binary to MOF

---

Input: n-bit binary string  $d = d_{n-1}|d_{n-2}|.....|d_1|d_0$

Output: MOF

$$md_n = d_{n-1}$$

for  $i = n - 1$  down to 1 do

$$md_i = d_{i-1} - d_i$$

$$md_0 = -d_0$$

Return  $(md_n, md_{n-1}, .....md_1, md_0)$

---

## 5.6 Proposed One's Complement Subtraction (OCS) Algorithm for Recoding of Scalar $k$

The concept of subtraction by utilization of complements is referred in literature since long time [87]. As subtraction by complement is performed in a manner that appears as addition, it permits the use of greatly simplified electronics circuits and is consequently very important in applied computation. A circuit whose only requirement is to perform addition is considerably simpler. Ideally, circuit should be as simple as addition circuit but capable of performing both addition and subtraction. This ideal situation is approached by performing subtraction by a special technique, i.e. addition of *numerical complements*. This section deals with the manner in which the *complementary technique* can be utilized in the decimal and binary system and then its use as recoding scalar with minimum Hamming weight in our proposed algorithms for WSN.

### 5.6.1 The 10's complement

The 10's complement of any number can be found by

$$C_{10} = 10^a - k$$

Where,

$C_{10}$  = 10's complement of any number  $N$ .

$k$  = Number whose complement is to found.

$a$  = Number of digits that can be handled by microcontroller.

**Example 5.1:** Subtract 240 from 380 by use of the 10's complement method. Assume 10 digits can be handled.

**Solution:** Find the 10's complement of 240.

$$\begin{array}{r} C_{10} = 10^a - 240 \\ 1000000000 \\ - \quad \quad 240 \\ \hline 9999999760 \end{array}$$

In order to facilitate the addition process, the desired numbers of 0's are tacked in front of the minuend (380). Now adding to obtain the remainder,

$$\begin{array}{r} 0000000380 \text{ Minuend} \\ + 9999999760 \text{ Subtrahend} \\ \hline 10000000140 \text{ Remainder} \end{array}$$

Notice the digit 1 in the  $11^{\text{th}}$  position. This indicates that the sum is positive. Since the sum is positive, the answer is 140.

If the sum is negative, in that case determine the 10's complement of the remainder and attach negative sign to it. That will be the answer.

The rules for subtraction by utilization of 10's complement can be summarized as per below-

1. Determine the 10's complement of the subtrahend.
2. Find the sum of the minuend and the 10's complement of the subtrahend.
3. If the sum has digit remainder as shown in the above example, the remainder is positive and the process is completed.
4. If there is no digit remainder, determine the 10's complement of the remainder, and attach negative sign to it.

### 6.5.2 The 9's Complement

The 9's complement of any number can be found by

$$C_9 = (10^a - 1) - k$$

Where  $C_9=9$ 's complement of the number  $N$ .

$k$  = Number whose compliment is to be found.

$a$  = Number of digits to be carried out in the complementing process.

**Example 5.2:** Determine the 9's complement of 270. Assume that 10 digits can be handled by the computer.

**Solution:**  $C_9 = (10^a - 1) - 270$

$$\begin{array}{r} 10000000000 \\ - \quad \quad \quad 1 \\ \hline 9999999999 \end{array}$$

### Step 1



Step2  $C_9 = 9999999999 - 270 = 9999999729$ .

Therefore the  $C_9$  of 270 is 9999999729.

### 6.5.3 Binary Subtraction By Utilization of 2's Complement

The rules for subtraction by use of 2's complement are identical to those used in the 10's complement process examined in the previous sections. Example 5.3 illustrates this for case in which the remainder is positive, while example 5.4 depicts subtraction where a negative remainder results.

**Example 5.3:** Subtract 0001110110 from 1101001011 by use of the 2's complement. Ten digits are to be used.

Solution:

Step 1: Find the 2's complement of the subtrahend.

$$\begin{array}{rcl}
 2^{10} & = & 10000000000 \\
 & - & 0001110110 \\
 & \hline
 & & 1110001010 = C2
 \end{array}$$

Step 2: Add the 2's complement of the subtrahend to the minuend using the rules of binary addition.

$$\begin{array}{r}
 1101001011 \\
 +1110001010 \\
 \hline
 11011010101 \\
 -
 \end{array}$$

Since the digit reminder is 1, the existing remainder is positive. Therefore 1011010101 is the remainder.

Checking the results by converting the binary quantities in to their decimal equivalent and subtracting,

$$\begin{array}{l}
 \text{Binary } 1101001011 = \text{Decimal } 843 \\
 \text{Binary } 0001110110 = \text{Decimal } 118 \\
 \text{Binary } 1011010101 = \text{Decimal } 725
 \end{array}$$

And subtracting directly in decimal,

$$\begin{array}{r}
 843 \\
 -118 \\
 \hline
 725
 \end{array}$$

**Example 5.4** Subtract 11011 from 10010 by utilization of the 2's complement technique. Assume the use of 10 digits.

Step 1: find the 2's complement of the subtrahend.

$$\begin{array}{r}
 2^{10} = 10000000000 \\
 - 0000011011 \\
 \hline
 1111100101 = C2
 \end{array}$$

Step 2: Add the 2's complement of the subtrahend to the minuend using the rules of binary addition.

$$\begin{array}{r}
 0000010010 \\
 + 1111100101 \\
 \hline
 0111111011
 \end{array}$$

Since the digit remainder is 0, the existing remainder must be recomplemented.

Step 3: Recomplement the remainder 111111011.

$$\begin{array}{r}
 2^{10} = 10000000000 \\
 - 1111110111 \\
 \hline
 0000001001
 \end{array}$$

Therefore the remainder is -1001.

Checking the decimal,

$$\begin{array}{r}
 10010 = 18 \\
 -11011 = 27 \\
 \hline
 -01001 = -9
 \end{array}$$

### 6.5.4 Binary Subtraction by Utilization of 1's Complement

A subtraction by utilization of the 1's complement is most common in binary arithmetic. The 1's complement of any binary number may be found by the following equation [88]:

$$C_1 = (2^a - 1) - k. \quad (I)$$

Where,  $C_1$  = 1's complement of the binary number.

$a$  = number of bits of  $k$  in terms of binary form.

$k$  = Integer in binary form.

In the case where computer will handle 10 digits :

$$C_1 = (2^{10} - 1) - k.$$

Note that

$$\begin{array}{r} 2^{10} = 10000000000 \\ - 0000000001 \\ \hline 1111111111 \end{array}$$

Therefore,

$$C_1 = 1111111111 - k$$

Notice that each digit of the binary number  $k$  is subtracted from 1. Recalling the rules for direct subtraction,

$$1 - 0 = 1$$

$$1 - 1 = 0$$

Examination of these two rules indicates that subtracting the digit from 1 actually changes the original digit. In effect, then, the 1's complement of any number can be obtained by changing all the 1s to 0s and all 0s to 1s. Microcontroller of WSN node can perform the 1's complement of conversion by use of inverter circuit or instruction COM. The inverter circuit 'flips' the signal from 0 input to 1 output or vice versa.

Subtraction by the 1's complement technique is performed by the following steps:

1. Determine the 1's complement of the subtrahend.
2. Add the 1's complement of the subtrahend to the minuend.
3. Perform the end around carry. If a 1 is involved in this process, the final remainder has been obtained. If a 0 is involved, proceed to step 4.
4. When the end around carry is with a 0, recomplement the remainder. This recomplemented remainder with a minus sign in front of it is final answer.

A close observation of the equation (I) reveals the fact that any positive integer  $k$  can be represented by using minimal non zero bits in its 1's complement form provided that it is having minimum of 50% Hamming weight.

The equation (I) can be modified as per below-

$$k = (2^a - C_1 - 1) \dots \dots \dots (II)$$

For example, let us take  $k = 1788$

$k = (11011111100)_2$  in its binary form

$C_1 = 1$ 's Complement of the number of  $k = (00100000011)_2$

Number of bits of  $k$  in binary form,  $a = 11$

After putting all the above values in the equation II we will get,

$$1788 = 2^{11} - 00100000011 - 1, \text{ this can be reduced to,}$$

$$1788 = 100000000000 - 00100000011 - 1$$

$$1788 = 100000000000 - (00100000011 + 1)$$

$$1788 = 100000000000 - (00100000100)$$

Now apply only bitwise subtraction rule of  $0 - 1 = \bar{1}$ .

$$1788 = 100\bar{1}00000\bar{1}00\dots\dots(III)$$

$$1788 = 2048 - 256 - 4$$

As evident from equation III the Hamming weight of integer  $k$  has reduced from 8 to 3 which will save 5 elliptic curve addition operations. One addition operation requires 2 Squaring, 2 Multiplication and 1 inverse operation. In this case total 6 Squaring, 6 Multiplication and 3 Inverse operations will be saved. The proposed method of One's Complement Subtraction (OCS) has given in the algorithm 5.2. OCS combined with sliding window method gives very good optimization results as compared to binary method.

---

**Algorithm 5.2 OCS algorithm proposed for recoding positive integer in WSN**

---

Input: A positive integer  $(k)_{10}$ .

Output: OCR of  $(k)_2 = \sum_{i=0}^l k_i 2^i, k \in \{0, \pm 1\}$ .

1. Convert  $(k)_{10} = (k)_2$

Where  $(k)_2 = \sum_{i=0}^{l-1} k_i 2^i, k \in \{0, 1\}$ . Also  $k_{l-1} \neq 0$ .

2. Obtain  $(\bar{k})_2$  by replacing

$(\bar{k}_i)_2 = 0$  if  $(k_i)_2 = 1$  and

$(\bar{k}_i)_2 = 1$  if  $(k_i)_2 = 0$ .

3. Obtain  $(k)_{OCR} = 2^l - (\bar{k})_2 - 1$ .

---

The OCS work further modified to TCS algorithm which is more computational friendly with WSN node using 8 bit microcontroller At mega 128L was developed and presented below.

### 6.5.5 Proposed TCS algorithm

---

**Algorithm 5.2 TCS algorithm for a positive integer**

---

Input: A positive integer in the TCS forms  $TCS(k)_2 = \sum_{i=0}^l k_i 2^i, k \in \{0, \pm 1\} \cdot (k)_{10}$ .

Output:  $TCS = 2^l > (k)_2 > 2^{l-1}$ .

Proof :

$$1. (k)_{OCS} = 2^l - (\bar{k})_2 - 1.$$

$$2. (k)_{OCS} = 2^l - [(\bar{k})_2 + 1]$$

$$3. (k)_{OCS} = 2^l - [2's \text{ complement of } (k)_2]$$

$$4. (k)_{TCS} = 2^l - [2^l - (k)_2]$$

---

## 5.7 Performance Evaluation of OCS algorithm on MATLAB

The following table gives performance evaluation of proposed method on MATLAB. The table also gives execution time for other methods namely Binary method , Non Adjacent Form (NAF) method , the Mutual Opposite Form (MOF) method for the same key sizes. We implemented our proposed new method on Intel P4 dual core processor with 1.8 GHz speed and 768 MB memory with MATLAB software. The MATLAB code for all the 4 methods is given below-



```

clc
    close all;
    clear all;
display('    program for Decimal to binary  ')
    display('=====')
    display('.'.')

k=input(' input integer number=====>')
tic
c=k
for k=1:1000
y=dec2bin(c)
z=mod(y,8)
p=1
%p=input( 'enter the value of p' )
q=0
for i=1:length(z)
    q=2*q
    if z(i)==1
        q=q+p
    end
end
end
toc
t=toc/1000

```

Figure 5. 2 MATLAB code for Decimal to Binary Conversion

Form the table we find that our proposed method takes least time as compared to other known recoding methods.

```

close all;
clear all;

display('    program for MOF')
display('=====')
display('.' )

k=input(' input any integer number =====>')
tic
    c=k

    for k=1:1000

        y=dec2bin(c);
        s=mod(y,8)
        x=s;
        s=[s 0]
        x=[0 x]
        z=s-x
        p=1
        %p=input('enter the value of p' )
        q=0
        for i=1:length(z)
            q=2*q
            if z(i)==1
                q=q+p
            else if z(i)==-1
                q=q-p
            end
        end
    end
end

toc
t=toc/1000

```

Figure 5. 3 MATLAB Code For Decimal to MOF Form

```

display('.'.
        k=input(' input any integer number =====>')
tic
c=k
for k=1:1000
s1=[]
for i=1
while (c>0),
    d=mod(c,2);
    if d==1
        m=mod(c,4);
        s(i)= 2-m;
        c=c-s(i);

    else
        s(i)=0;

    end
    c=c/2;
    i=i+1;
end
s1=[s1 s];
z=fliplr(s1)
end
p=1
%p=input( 'enter the value of p' )
q=0
for i=1:length(z)
    q=2*q
    if z(i)==1
        q=q+p
    else if z(i)==-1
        q=q-p
    end
end
end
end
toc
t=toc/1000

```

Figure 5. 4 MATLAB Code For Decimal To NAF Form

```

clc
clear all;
display('      program for CRM')
display('=====')
display('.'
        k=input(' input any integer number =====>')
tic
        c=k;

        for k=1:1000
            y=dec2bin(c);
            x=mod(y,8)
            p=~(x)
            q=length(y);
            r=[1 zeros(1,q)]
            a=[0 p]
            b=r-a;
            f=length(b);
            d=[zeros(1,f-1) 1];
            z=b-d
            p=1
%p=input('enter the value of p')
q=0
for i=1:length(z)
    q=2*q
    if z(i)==1
        q=q+p
    else if z(i)==-1
        q=q-p
    end
end
end
        end
toc
t=toc/1000

```

Figure 5. 5 MATLAB code for OCS Algorithm

```

display('      program for PM')
display('=====')
display('.' )

      k=input(' input any integer number =====>')
tic

      c=k;
      for k=1:1000

          for p=1:inf,

              x= (2^p);
              y=c-x;
              if (y<=0),
                  r=x
                  break;

          end
      end

      q=dec2bin(r);
      q=mod(q,8)
      q1=length(q);
      s=r-c;
      v=dec2bin(s);
      v=mod(v,8)
      v1=length(v);
      u=[zeros(1,(q1-v1)) v];
      z=q-u
      p=1
      %p=input(' enter the value of p' )

```

Figure 5. 6 MATLAB code for TCS algorithm

Table 5. 1 Comparison of OCS with other methods on MATLAB

| Bit Size | Binary Method | NAF method | MOF Method | OCS Method |
|----------|---------------|------------|------------|------------|
| 26       | 17.12         | 15.80      | 13.80      | 11.7       |
| 36       | 20.12         | 19.42      | 17.7       | 15.10      |
| 43       | 23.00         | 22.00      | 19.28      | 17.28      |
| 52       | 25.90         | 23.20      | 20.20      | 19.35      |

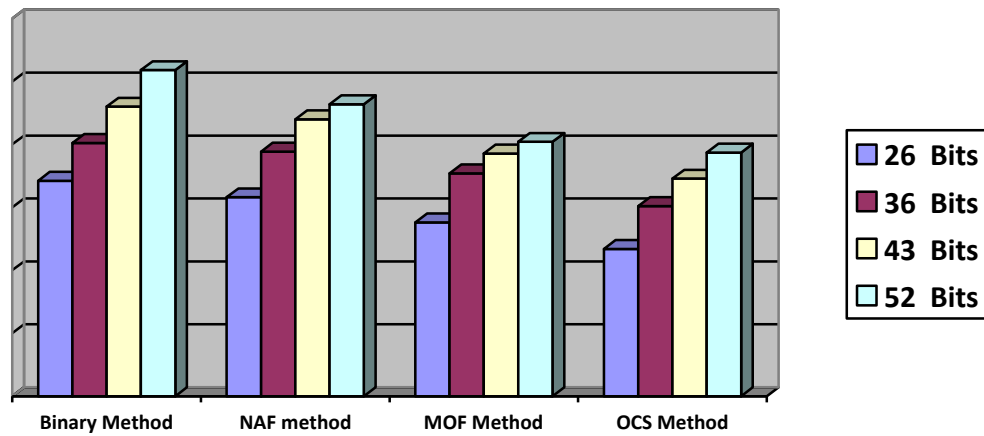


Figure 5. 7 Comparison of OCS with other algorithms on MATLAB

## 5.8 Summary

The positive integer in point multiplication can be recoded with *one's complement subtraction* (OCS) to reduce the computational cost involved in this heavy mathematical operation for WSN platform. The window size may be a subject of trade off between the available RAM and ROM at that particular instance on sensor node. As NAF method involves modular operations to get the NAF of binary number, the OCS and TCS can provide a very simple way of recoding integer.

## Chapter 6 Proposed Window OCS Algorithm for Prevention of SPA in WSN

---

### 6.1 Introduction

All ECC protocols are based on *point addition* and *point doubling* operations. These two operations require different power and execution time on WSN node. In binary method of scalar multiplication, point addition always corresponds to 1 and point doubling corresponds to 0. The private key of the WSN node is recoded in the form of 1 and 0, while doing scalar multiplication. All this knowledge in the public domain provides sufficient *side channel leakage* at lower levels to the attacker to know the entire key in the sequence of 1 and 0 by analyzing power consumption of microcontroller and time required for execution on WSN node for particular cryptographic routine as shown in Figure 6.1. These types of attacks are not theoretical and can be carried out with instruments such as power oscilloscopes to measure the power consumption of sensor nodes while implementing cryptographic algorithm. Figure 6.2 shows setup of SPA attack.

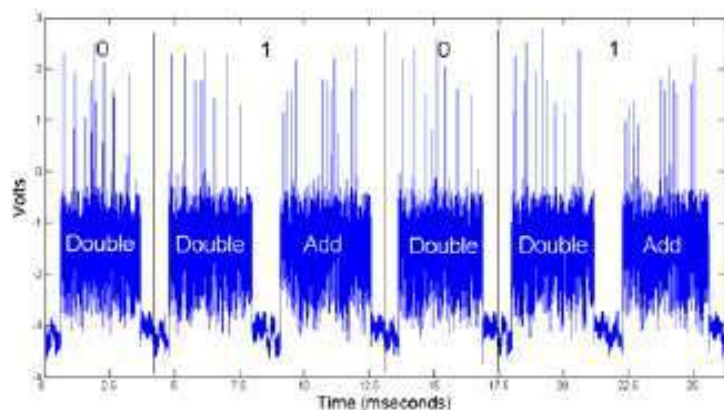




Figure 6. 1 Power traces revealing value of private key of the WSN node [63]

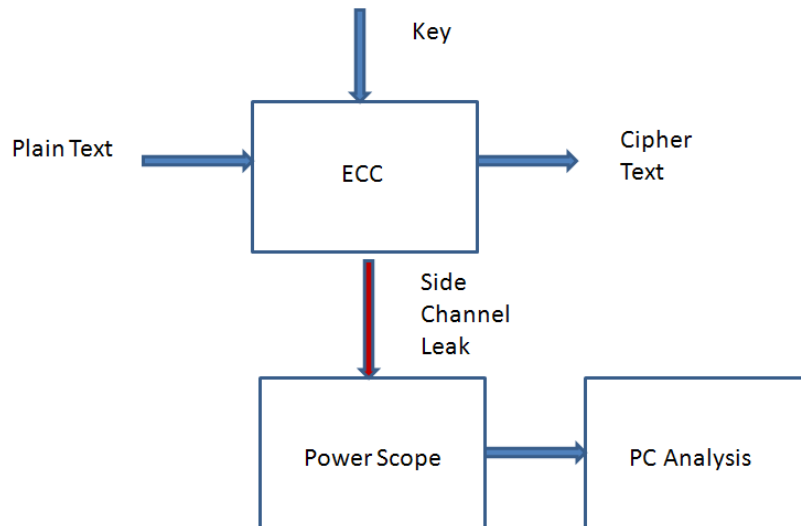


Figure 6. 2Block diagram of SPA setup

As shown in figure 6.1 point doubling and point addition power spikes are distinguishable from each other. Next section gives detailed analysis of basic building block of microcontroller i.e. CMOS inverter circuit to understand the nature of power analysis attacks.

## 6.2 Overview of side channel attacks on WSN node

There are several method available for obtaining the side channel information in order to find the secret key in ECC. This chapter deals with only SPA, however several other methods of attack are also briefly summarized here for completeness.

### 6.2.1 CMOS Inverter Circuit and Simple Power Analysis

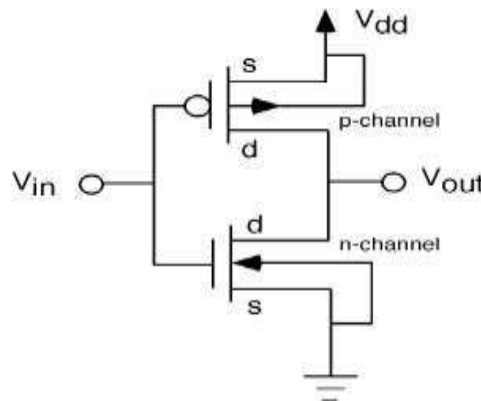


Figure 6. 3CMOS Logic Inverter Circuit

CMOS inverter circuit consists of two transistors namely P-channel and N-channel that serves as semiconductor switches and turns on or off depending on the input voltage  $V_{in}$ . The input to the transistor may be at logic 1 or logic 0. Logic 1 is called the high voltage signal and logic 0 is called low voltage signal. If the  $V_{in}$  is at logic 1, then P-channel transistor is non conducting and N- channel transistor is conducting. In this case there current will flow from output to the ground and output voltage will be logic 0 or equal to ground voltage. If the  $V_{in}$  is at low voltage signal in that case P-channel transistor is conducting and N-channel is non conducting and the supply voltage  $V_{dd}$  will appear at the output terminal and output will be will be high i.e. logic 1. So this invert circuit gives output 1 if the input is 0 and vice versa.

The power consumption of the above circuit during each clock cycle can be measured by placing a standard resistor of value  $1\ \Omega$  value in series with supply voltage  $V_{dd}$ . This power depends on the instruction being executed and someone can plot a power trace, which shows power consumed by device during each clock cycle.

The hypothesis behind power analysis attacks is that the power traces are always correlated to the instructions the microprocessor is executing as well as the value of the operands it is manipulating [57].

Therefore, examinations of the power traces can reveal information about the instructions being executed and contents of the data registers. In the case that the device is executing as secret key cryptographic operation, it may be possible to deuce the secret key. The figure 6.4 shows set up of SPA in WSN.

Several authors have proposed unified code to avoid this weakness which eliminates the timing differences of point addition and point doubling operations. Unfortunately these techniques are difficult to implement on sensor nodes due to limited memory and computational power. This research proposes an innovative algorithm which is light weight and can be implemented at physical layer of wireless sensor nodes to avoid any information leakage.

Simple power analysis (SPA) attacks were proposed by[89] .A more sophisticated use of SPA were proposed by [90, 91] where the attacker examines the Hamming weight of the positive integer by measuring height of the power trace at a point during scalar multiplication process. By observing hamming weight of private key it is possible to determine the private key used by WSN node.

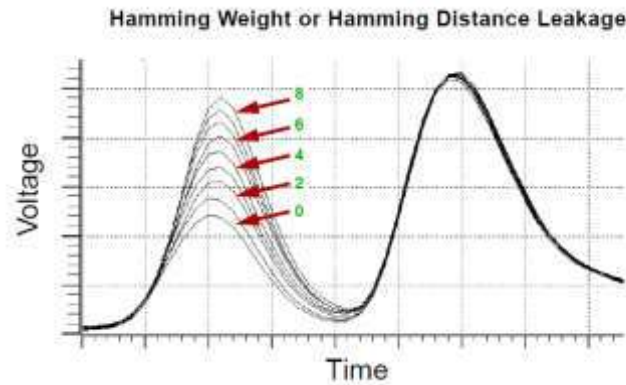


Figure 6. 4Example of Power Consumption Information Leakage [90]

### 6.2.2 Enhanced Simple Power Analysis

In scalar multiplication if the integer is recoded with signed binary digits, addition subtraction method is used to generate the public key.[92] proposed enhanced SPA which makes use of Markov Chain to predict the bits of the private key which have caused particular pattern of point additions and doublings operation. This method can also be applied to randomized addition-subtractions chain models.

### 6.2.3 Differential Power Analysis

These attacks were proposed by [89] .These attacks uses a statically analysis of power signals from many scalar multiplication process to examine the similarity and differences between these power traces. These similarities and differences are due to various values of data, operands and register addresses used during programming. This information is then manipulated to reveal the secret key of the node.

Various counter measures are available in literature for DPA. These counter measures includes,

- 1.Randamization of the private key of the node.
- 2.Use of addition mask.

3. Use of exponent splitting.
4. Overlapping window method.
5. Randomised table window method.
6. Hybrid overlapping and randomized table window method.
7. Perturbation point.
8. Point of small order as a perturbation point.
9. Randomized projective or Jacobian coordinates.
10. Randomized Isomorphic Curve or field
11. Scalar Multiplication Algorithms Randomization

The wireless sensor nodes like IMOTE2 , MICAz [21] are manufactured by using CMOS (Complementary Metal Oxide Semiconductor) technology in which the basic building block is the inverter or NOT gate as depicted in Figure 7.1.

#### **6.2.4 Electro magnetic Analysis**

The flow of current through CMOS device also induces electromagnetic radiations. This EM signal can be collected by placing sensor close to the device. As with power analysis attacks, one can now analyze the EM signal to reveal the secret key. Simple Electro Magnetic Analysis, Differential Electro Magnetic Analysis can be launched. As with power analysis counter measures for these attacks could be hardware based, for e.g. use of metal layer or circuit designing etc.

### 6.2.5 Fault and Timings Attacks

Fault attacks are always carried out by feeding invalid input or causing fault during as computation by the microcontroller of WSN node. The output of the microcontroller in such circumstances can reveal the information about the key. Timings attacks work when an operation using secret key does not run in the constant time and the time taken is correlated to the value of the secret key.

## 6.3 Overview of Existing SPA counter Measures

There are two standard approaches against SPA attacks: first one is use of dummy operations and second one is use of identical formulae. Both approach attempts to make the power traces of the two group operations (addition and doubling) indistinguishable [30]. The first approach consists in adding extra or “dummy” operations in the addition and doubling algorithms where the sequences of operations differ. In the second approach the same sequence of operations will be repeated independent of scalar, so they will appear identical to SPA attacks. The first approach will increase the code size and will put microcontroller in idle time where as second approach will increase the cost of scalar multiplication algorithm.

Although dummy operation is a very simple countermeasure to implement, it is not always safe: If the secret key is used multiple times, dummy operations can be revealed by adaptive fault analysis, and further countermeasures are required to prevent this attack [30].

The second approach consists in rewriting the two group operations into a unified formula. Since both operations will then use the same set of operations, the two operations will have the same power trace. Unified formulae tend to be more costly to use than dummy operations, but they prevent adaptive fault analysis (but not DPA). The main disadvantage of unified formulas is that they are group specific and so far they have only been developed for elliptic curves. The following section gives various counter measures available for SPA in the literature.

### 6.3.1 Double and Add Always

This method modifies binary method slightly to achieve resistance against SPA. Instead of performing A point addition when the next bit of the integer k is 1, this algorithm performs an addition in each iteration of the loop and discards the results of those additions which are irrelevant. This prevents SPA because the pattern of additions and doublings is the same for all scalar multiplications and does not reveal the integer used.

#### Algorithm 6. 1 SPA Resistant Double and Add Always Method

---

#### Algorithm 6.1 SPA Resistant Double and Always Method

---

*Input :* Point  $P \in E(F_q)$ , an  $\ell$  bit integer  $k = \sum_{j=0}^{\ell-1} k_j 2^j, k_j \in \{0,1\}$

*Output :*  $Q = kP$

1.  $Q \leftarrow \infty$ . For  $j = \ell - 1$  to 0 do,

1.1  $Q_0 \leftarrow 2Q_0$ ,

1.2  $Q_1 \leftarrow Q_0 + P$

1.3  $Q_0 \leftarrow Q_{k_j}$

2. Return ( $Q_0$ )

---

### 6.3.2 Montgomery Ladder

A Montgomery ladder [26] can be used to resist SPA because it performs a double and add in every loop, in a similar fashion to the double and add always countermeasures. A Montgomery ladder does not use any dummy operations. The method was originally proposed for use with Montgomery curves, since it allowed a fast point addition algorithm

which did not require the use of the y-coordinate on these curves. A version of the Montgomery ladder which was described in [26] is as shown below-

**Algorithm 6. 2 Montgomery Ladder to avoid SPA [26]**

---

**Algorithm 6.2 Montgomery Ladder to avoid SPA**

---

Let  $k = k_{u-1} \dots k_1 k_0$  be the secret integer and suppose we need to find  $k[P]$ .

Let  $P_1 = P, P_0 = [2]P$ .

For  $i$  from  $u - 2$  down to  $0$  do:

- $j = 1 - k_i$
- $P_j = P_0 + P_1$
- $P_{k_i} = [2]P_{k_i}$

**Return**  $P_1$

---

A method for finding the addition and doubling required by Montgomery's ladder without using the y-coordinate of the points for any elliptic curve has been proposed by [93].

### 6.3.3 Identical formulae for point addition and doublings

[93] proposed a new method in which point doubling and point addition will have the same formulae for any curve over prime field without using Montgomery ladders. Formulae for addition and doubling in both affine and projective coordinates have been provided.

The other methods to avoid SPA attacks includes universal exponentiation algorithm, randomized addition and subtraction chain, non deterministic right to left method with pre computations. All these methods which are suggested in literatures are either increasing computational cost or putting more stress on the memory usages to avoid SPA attacks. As seen in first few chapters the major objective of this research is to reduce the computational cost and memory usages for WSN and makes these counter measures heavy weight for them.



## 6.4 Proposed Window OCS method to avoid SPA in WSN

Our proposed algorithm is based on a *window principle* in which point doubling and addition operations will be performed at a consistent sequence independent of scalar. The fixed window method which is sometimes referred as *m-ary* method satisfies this requirement of consistency of point addition and doubling operation which is independent of scalar. The proposed *window OCS* method is a variant of window method in which the scalar  $k$  is represented in the OCS from by use of algorithm 5.2, chapter 5 of the thesis. The use of window method as countermeasure against SPA was initially proposed by Bodo Moller [27]. Our algorithm is different from Bodo's algorithm because it uses different *encoding* method for scalar than Bodo's method. The proposed algorithm based on windowing principle is effective method on WSN platform as this method do not use any dummy operations and does not limit to particular family of curves and thus can be implemented on any NIST curves. Proposed *window OCS* method resistant SPA is given in the algorithm 6.3.

---

Algorithm 6.3 Proposed Window OCS method for scalar multiplication

---

*Input : Window width  $w$ , a positive integer  $k$ , point  $P$ .*

*Output : Scalar Resistant  $k[P]$*

1. Use algorithm 5.2 to compute OCS of  $k = \sum_{i=0}^{l-1} k_i 2^i, k \in \{\pm 1, 0\}$

2. Compute  $P_i = iP$  for  $i \in \{1, 3, 5, \dots, 2^{w-1} - 1\}$ .

3.  $Q \leftarrow \infty$ .

*Main Loop.*

4. For  $i$  from  $l-1$  to 0 do :

4.1.  $Q \leftarrow 2Q$ . *Doubling*

4.2. If  $k_i \neq 0$  then

If  $k_i > 0$ , then  $Q \leftarrow Q + P_{k_i}$ . *Addition*

Else  $Q \leftarrow Q - P_{k_i}$  *Subtraction*

7. Return ( $Q$ ).

---

As shown in algorithm 6.3, window OCS method with a window size of  $w$  requires pre-computing of the points  $2P, 3P, \dots, (2^w - 1)P$ . These  $2^w - 2$  points are stored in a look up table, typically in affine representation to save RAM and to allow one using the mixed coordinates for point addition as discussed in the Chapter 4 of this thesis. Our OCS window algorithms works in a similar fashion as the double and add method, except that in each step  $w$  bits of  $k$  are considered with the corresponding table entry being added to the intermediate results. A window size of  $w$  reduces the total number of additions to  $l/w$  where  $l$  are the total number of bits in  $k$ , but does not change the number of doublings ( $l$  doublings). The next chapter of this thesis aims at finding out optimal value of window size  $w$  of window OCS

method so that there will be good compromise between performance and memory requirements of WSN node to avoid SPA.

The window OCS method employs a “sliding window” in the sense that algorithm 6.3 has a width of  $w$ , moving right to left, skipping consecutive zero entries after a non zero digit  $k_i$  is processed. Although we have found no reference to this specific scheme in the literature, a suggestion to combine m-ary and signed digit method appears in [27]

## 6.5 Summary

The WSN node consists of limited memory for storage so to avoid SPA attack is very difficult in such environment. Our window OCS method based on windowing principle will execute point addition and doubling in uniform sequence irrespective of scalar and will make attacker difficult to detect the secret scalar by using SPA. The window size  $w$  is a matter of trade off between the available memory and performance. Next chapter of the thesis will present innovative algorithm to choose the optimum choice of window  $w$  for WSN so that there is balanced use of memory for application and computational purposes and was a major objective of this research.

## Chapter 7 Proposed Elastic Window Method of Scalar Multiplication for WSN

---

### 7.1 Introduction

The basic building blocks of an ECDH, ECDSA and other standards elliptic curve protocols is scalar multiplication which can be given by formula,

$$Q = [k]P = \underbrace{P + P + \dots + P}_{k \text{ times}}$$

where  $P$  is a point on the elliptic curve, and  $k$  is a positive integer in the range  $1 \leq k < \text{ord}(n)$ . For Some of the cryptographic protocols,  $P$  is a fixed point that generates a large, prime order subgroup of  $\#E(F_q)$ , while for others  $P$  is dynamic point in such a subgroup. The strength of the cryptosystem lies in the fact that given the curve, the point  $P$  and  $[k]P$ , it is hard to recover  $k$ . This is called the Elliptic Curve Discrete Logarithm Problem (ECDLP) and the computation  $[k]P$  referred as scalar *multiplication* or *point multiplication*.

To optimize and accelerate scalar multiplication process on WSN is the major objective of this thesis as discussed in the Research problem in Chapter 1. This chapter gives overview of existing methods of scalar multiplication methods and proposes *Elastic window method* which is a variant of sliding window method in which window size  $w$  will be selected based on available RAM of WSN node so that there will not be WSN node failure due to stack overflow problem.

## 7.2 Overview of Existing Methods of Scalar Multiplication

Scalar multiplication in elliptic curves is a special case of the general problem of exponentiation in abelian groups. Efficient algorithms for group of exponentiation have received much attention by researchers in recent years, owing to their central role in the public key cryptography. The interest in the problem, however, is ancient. An excellent technical and historical account of exponentiation and the addition chain problem is given by Knuth, who traces the problem back to 200BC. The survey of Gordon describes various fast methods, including some specialized to elliptic curves groups. The major points to remember are,

1. Elliptic curve subtraction has virtually the same cost as addition.
2. Modular inversion is highly expensive operation.
3. For certain families of elliptic curves, specific shortcuts are available that can significantly reduce the computational cost of point multiplication.
4. For the sake of concreteness, when analyzing computational complexity, we will focus on the case of finite field of characteristic two. Also, for simplicity, we will neglect the cost of squaring in these fields.

### 7.2.1 The Binary Method for Scalar Multiplication

The simplest and oldest method for point multiplication relies on the binary expansion of  $k$ . The cost of scalar multiplication in binary method depends on the number of non-zero elements and length of the binary representation of  $k$ . If the representation has  $k_{l-1} \neq 0$ , then

binary method require  $(l-1)$  point doubling and  $(H-1)$  where  $l$  is the length of the binary expansion and  $H$  is the Hamming weight of the  $k$  (the number of non-zero elements).

For example,

if  $k = 629 = (1001110101)_2$ , it will require  $(H-1) = 6-1 = 5$  point additions and  $(l-1) = 10-1 = 9$  point doublings.

The binary method for the computation of scalar multiplication is given in the Algorithm 5.1 and 5.2. Algorithm 5.1 processes bits of  $k$  from left to right and Algorithm 5.2 processes bits from right to left.

All the previous researchers have preferred binary method for scalar multiplication as it does not require extra memory to store the pre-computed values. It scans a one bit at time from right to left or vice versa. But as discussed in chapter 6 it is susceptible to SPA attacks.

### 7.2.2 The $m$ -ary Method

This method uses the  $m$ -ary expansion of  $k$  where  $m = 2^r$  for some integers  $r \geq 1$ . The binary method is a special case corresponding to  $r = 1$ .

#### Algorithm 7.1 Multiplication by m-ary method

---

##### Algorithm 7.1 Multiplication by m-ary method

---

*Input* : A point  $P$ , an integer  $k = \sum_{j=0}^{d-1} k_j m^j, k_j \in \{0, 1, \dots, m-1\}$ .

*Output* :  $Q = [k]P$ .

*Precomputation.*

1.  $P_1 \leftarrow P$ .

2. For  $i = 2$  to  $m-1$  do  $P_i \leftarrow P_{i-1} + P$ . ( $P_i = [i]P$ ).

3.  $Q \leftarrow \theta$ .

*Main Loop.*

4. For  $j = d-1$  to  $0$  by  $-1$  do :

5.  $Q \leftarrow [m]Q$ . (This requires  $r$  doublings.)

6.  $Q \leftarrow Q + P_{k_j}$ .

7. Return  $Q$ .

---

### 7.2.3 Sliding Window Method

If some extra memory is available the performance of binary method can be improved by scanning few bits at a time as with *sliding window method*. This method processes a window of length  $w, w > 1$ , disregarding fixed digit boundaries, and skips runs of zeros between them as shown in Algorithm 7.2 [4]. These runs are taken care by point doubling.

---

Algorithm 7.2 Sliding Window method

---

*Input* : A point  $P$ , an integer  $k = \sum_{j=0}^{\ell-1} k_j 2^j, k_j \in \{0,1\}$ .

*Output* :  $Q = [k]P$ .

*Precomputation.*

1.  $P_1 \leftarrow P, P_2 \leftarrow [2]P$ .

2. For  $i = 1$  to  $2^{w-1} - 1$  do  $P_{2i+1} \leftarrow P_{2i-1} + P_2$ .

3.  $j \leftarrow \ell - 1, Q \leftarrow \theta$ .

*Main Loop.*

4. While  $j \geq 0$  do :

5. If  $k_j = 0$  then  $Q \leftarrow [2]Q, j \leftarrow j - 1$ .

6. Else do :

7. Let  $t$  be the least integer such that

$j - t + 1 \leq w$  and  $k_t = 1$ ,

8.  $h_j \leftarrow (k_j k_{j-1} \dots k_t)_2$ ,

9.  $Q \leftarrow [2^{j-t+1}]Q + p_{h_j}$ ,

10.  $j \leftarrow t - 1$ .

11. Return  $Q$ .

---

For the above algorithm the number of pre-computations required for unsigned binary number is given by  $2^{w-1}$  and for signed binary number are given by  $2^{w-1} - 1$  where  $w$  is the size of window and is greater than 1. The size of  $w$  may be selected on the basis of RAM available with WSN node. More is the size of  $w$ , faster will be the scalar multiplication process. But we can not increase the size of  $w$  beyond particular limit as it causes WSN node failure due to stack overflow problem. So optimizing the size of  $w$  is major objective of this chapter. We explain above phenomenon with example.



Let us compute  $Q = 763[P]$  with sliding window algorithm with  $k = 763$  recoded in binary form. Let us choose window size  $w$  ranging from 2 to 10.

| Let us compute $Q = [763] P$ with Sliding window with window size $w$ ranging from 2 to 10<br>$763 = [1011111011]_2$  |  |
|---|--|
| <ul style="list-style-type: none"> <li>Window Size <math>w = 2</math></li> <li>No of pre-computations = 2</li> <li>2P, 3P</li> <li><math>763 = 10 \ 11 \ 11 \ 10 \ 11</math> [The fonts indicated with brown colour shows window boundary]</li> <li>The intermediate values of <math>Q</math> are ,</li> <li>2P 4P, 8P, 11P, 22P, 44P, 47P, 94P, 188P, 190P, 380P, 760P, 763P</li> <li>Computational cost = 9 doublings, 4 additions, and 2 precomputations.</li> </ul> |  |
| <ul style="list-style-type: none"> <li>Window Size <math>w = 3</math></li> <li>No of precomputations = 4</li> <li>2P, 3P, 5P, 7P</li> <li><math>763 = 101 \ 111 \ 101 \ 1</math></li> <li><math>= 5P \ 7P \ 5P \ 1P</math></li> <li>The intermediate values of <math>Q</math> are ,</li> <li>5P, 10P, 20P, 40P, 47P, 94P, 188P, 376P, 381P, 762P, 763P</li> <li>Computational cost = 7 doublings, 3 additions, and 4 precomputations.</li> </ul>                        |  |
| <ul style="list-style-type: none"> <li>Window Size <math>w = 4</math></li> <li>No of precomputations = 8</li> <li>2P, 3P, 5P, 7P, 9P, 11P, 13P, 15P</li> <li><math>763 = 1011 \ 111 \ 0 \ 11</math></li> <li><math>= 11P \ 7P \ 3P</math></li> <li>The intermediate values of <math>Q</math> are</li> <li>11P, 22P, 44P, 88P, 95P, 190P, 380P, 760P, 763P</li> <li>Computational cost = 6 doublings, 2 additions, and 8 precomputations.</li> </ul>                     |  |

|   |
|---|
| <ul style="list-style-type: none"> <li>Window Size <math>w = 5</math></li> <li>No of precomputations = 16</li> <li>2P, 3P, 5P, 7P, 9P, 11P, 13P, 15P, 17P, 19P, 21P, 23P, 25P, 27P, 29P, 31P</li> <li>763 = 10111 11011</li> <li>= 23P 27P</li> <li>The intermediate values of Q are</li> <li>23P, 46P, 92P, 184P, 368P, 736P, 763P</li> <li>Computational cost = 5 doublings, 1 additions, and 16 precomputations</li> </ul>   |
| <ul style="list-style-type: none"> <li>Window Size <math>w = 6</math></li> <li>No of precomputations = 32.</li> <li>2P,3P, 5P, 7P, 9P,11P,13P,15P,17P,19P, 21P, 23P, 25P, 27P, 29P, 31P, 33P, 35P, 37P, 39P, 41P, 43P, 45P, 47P, 49P, 51P, 53P, 55P, 57P, 59P, 61P</li> <li>763 = 101111 1011</li> <li>The intermediate values of Q are</li> <li>47P, 94P, 188P, 376P, 752P, 763P</li> <li>Computational cost = 4 doublings, 1 additions, and 32 precomputations</li> </ul>   |
| <ul style="list-style-type: none"> <li>Window Size <math>w = 7</math></li> <li>No of precomputations = 64</li> <li>2P,3P, 5P, 7P, 9P,11P,13P,15P,17P, 19P, 21P, 23P, 25P, 27P, 29P, 31P, 33P, 35P, 37P, 39P, 41P, 43P, 45P, 47P, 49P, 51P, 53P, 55P, 57P, 59P, 61P, 63P, 65P, 67P, 69P, 71P, 73P, 75P, 77P, 79P, 81P, 83P, 85P, 87P, 89P, 91P, 93P, 95P, 97P, 99P, 101P, 103P, 105P, 107P, 109P, 111P, 113P, 115P, 117P, 119P, 121P</li> <li>763 = 1011111 011</li> <li>The intermediate values of Q are</li> <li>95P, 190P, 380P, 760P, 763P</li> <li>Computational cost = 3 doublings, 1 additions, and 64 precomputations.</li> </ul>  |
| <ul style="list-style-type: none"> <li>Window Size <math>w = 8</math></li> <li>No of precomputations = 128</li> <li>2P,3P, 5P, 7P, 9P, 11P,13P,15P, 17P, 19P, 21P, 23P, 25P, 27P, 29P, 31P, 33P, 35P, 37P, 39P, 41P, 43P, 45P, 47P, 49P, 51P, 53P, 55P, 57P, 59P, 61P, 63P, 65P, 67P, 69P, 71P, 73P, 75P, 77P, 79P, 81P, 83P, 85P, 87P, 89P, 91P, 93P, 95P, 97P, 99P, 101P, 103P, 105P, 107P, 109P, 111P, 113P, 115P, 117P, 119P, 121P, 123P, 125P, 127P, 129P, 131P, 133P, 135P, 137P, 139P, 141P, 143P, 145P, 147P, 151P, 153P, 155P, 157P, 159P, 161P, 163P, 165P, 167P, 169P, 171P, 173P, 175P, 177P, 179P, 181P, 183P, 185P, 187P, 189P, 191P, 193P, 195P, 197P, 199P, 201P, 203P, 205P, 207P, 209P, 211P, 213P, 215P, 217P, 219P, 221P, 223P, 225P, 227P, 229P, 231P, 233P, 235P, 237P, 241P, 243P, 245P, 247P, 249P, 251P, 253P, 255P</li> <li>763 = 1011111 0 11</li> </ul> |

|  |
|--|
| <ul style="list-style-type: none"> <li>• The intermediate values of <math>Q</math> are</li> <li>• 95P, 190P, 380P, 760P, 763P</li> <li>• Computational cost = 3 doublings, 1 additions, and 128 precomputations.</li> </ul>  |
| <ul style="list-style-type: none"> <li>• Window Size <math>w = 9</math></li> <li>• No of precomputations = 256</li> <li>• 763 = 101111101 1</li> <li>• The intermediate values of <math>Q</math> are</li> <li>• 381P, 762P, 763P</li> <li>• Computational cost = 1 doublings, 1 additions, and 256 precomputations.</li> </ul> |
| <ul style="list-style-type: none"> <li>• Window Size <math>w = 10</math></li> <li>• No of precomputations = 512</li> <li>• = 1011111011</li> <li>• The intermediate values of <math>Q</math> are</li> <li>• 763P</li> <li>• Computational cost = 0 doublings, 0 additions, and 512 precomputations.</li> </ul>                 |

As shown in the above example, it is observed that as the window size  $w$  increases, the number of pre-computations also increases geometrically. At the same time number of additions and doubling operations decreases. But the question is whether WSN node is having that much memory to store all these pre-computed values? The trade-off between the computational cost , required memory and the window size  $w$  is shown in the Figure 5.8 and 5.9.

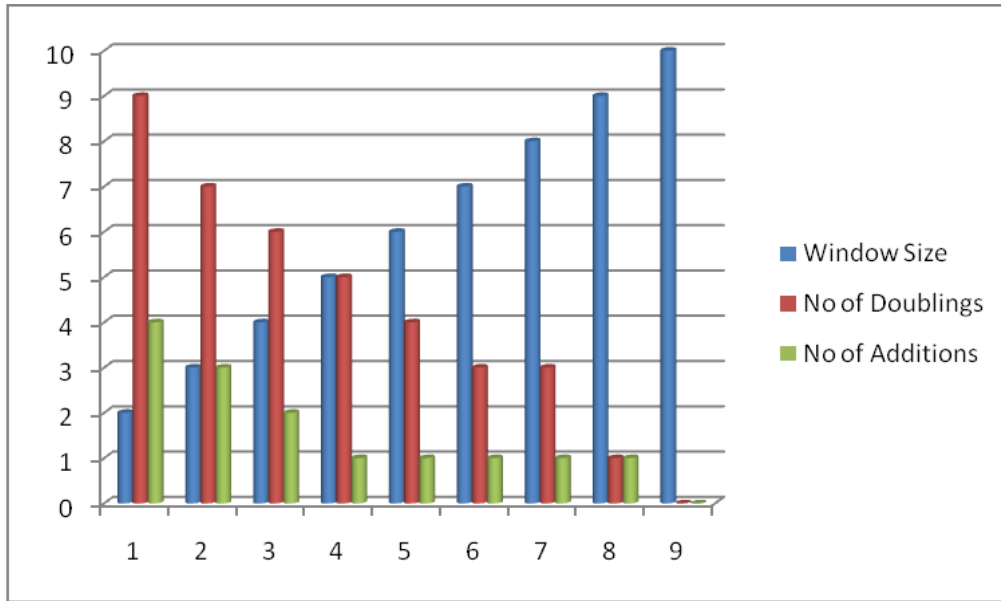


Figure 5. 8 Trade Off Between Window Size  $W$  and Computational Cost

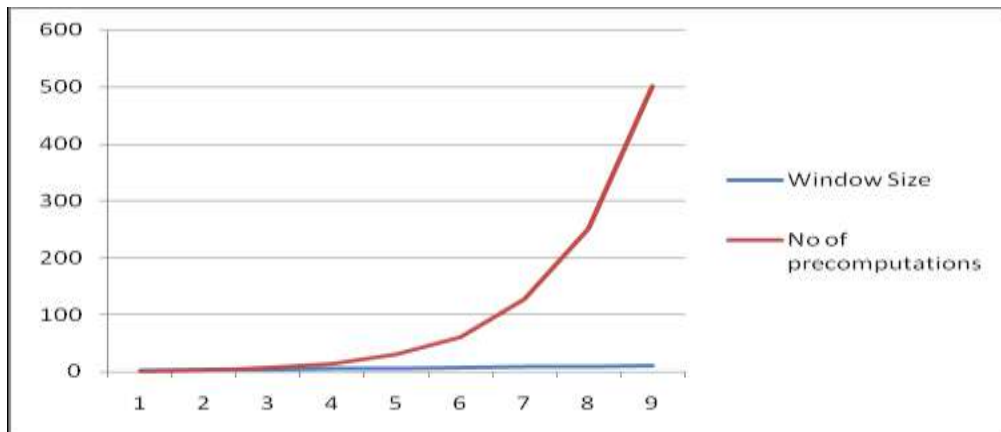


Figure 5. 9 Trade off between window size  $W$  and number of pre-computations

As a matter of interest, let us apply the proposed OCS algorithm in the chapter 5, to the same value of  $k = 763$  to show the effectiveness of OCS algorithm . Let us assume  $w = 3$  .

---

### Example 5.3

---

$$k = 763 = (1011111011)_2.$$

Let us recode  $k$  with OCS algorithm.

$$(763)_{OCS} = 10000000000-0100000100-1$$

$$= 10-100000-10-1.$$

Let us group by  $w = 3$ , by skipping consecutive zeros after 1.

$$= \text{10-100000-10-1.} \quad (\text{Window sizes is shown by brown fonts and negative number shown by - sign})$$

$$= \text{3P} \quad \text{-5P}$$

$$= 3P, 6P, 12P, 24P, 48P, 96P, 192P, 384P, 768P, 763P.$$

---

With the proposed OCS algorithm computational cost has been reduced from 3 additions as to only 1 addition as compared to normal window method with  $w = 3$ . The signed digit recoding method of OCS gives better results for WSN platform provided the window size has to be chosen appropriately. The next section gives effect of window sizes on memory of WSN nodes.

### 7.3 Effect of Window Size on Memory Utilization of Mica Nodes

It is easy to see that the sliding window method will increase both the ROM [for additional code size] and RAM [for storing the pre-computed points] consumptions of WSN nodes. The

Table 1 and 2 gives performance results which were measured on MIRACL for sliding window method for window size of 2 and 4. (Table to be included later).

Sliding window method can make signature generation and verification 1.2 times faster at the cost of dramatic RAM increases. Since MICAz, TelosB and Tmote Sky are basic WSN platforms, they have much smaller RAM (4kB, 10kB) compared with Imote2 (256kB). Before using sliding window method, we should be very careful if the sensing application has large RAM consumption.

Calling a function or handling an interrupt in WSN nodes causes stack memory to be allocated. In our case it is the function written for *scalar multiplication* by using sliding window method. When this interrupt occurs, the micro controller of WSN nodes allocates stack memory to store pre-computational values for scalar multiplication. If the stack memory region is not large enough to hold these values, a stack overflow occurs. Stack overflow leads to corrupted RAM and subsequently to non-deterministic WSN node failures. If the window size is too much large it may end up in node failure due to above reasons.

## 7.4 Stack Depth Analysis

A TinyOS on WSN node has a single stack and its size can be computed like this: Stack region size = RAM size – [Data Segment Size + BSS segment size]. In the most basic case, the memory model for a TinyOS application on a MICA 2 or MICAz platform with 4 KB of RAM looks like in Figure 7.1. The size of each region is immediately apparent. The only remaining question is: Is the stack memory region large enough to contain the actual stack and to hold pre-computed values of sliding window method?

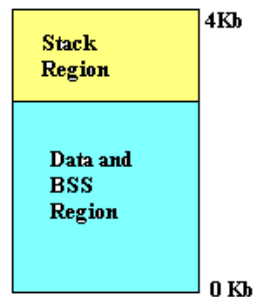


Figure 7. 1 Memory model for Tiny OS on MICA mote

Stack Depth Analysis can be done with following command-

```
[regehr@babel BaseStation]$ tos-ramsize micaz ./build/micaz/main.exe.
```

The result should be something like:

BSS segment size is 1708, data segment size is 16

The upper bound on stack size is 538

The upper bound on RAM usage is 2262

There are 1834 unused bytes of RAM

Here output is indicating that for application, approximately 1.8 KB of RAM is free and could have been allocated for pre-computations.

## 7.5 Proposed Elastic window Algorithm for scalar multiplication

This new algorithm makes sure that the chosen window size  $w$  of the sliding window method for scalar multiplication is *stack safe* and also make sure that memory is sufficient to hold all

the pre-computational values. Due to this reason its name is Elastic window. If the memory available is less, it will decrease window size  $w$  and if memory available is more it will increase size of  $w$  by making use of stack depth analysis.

**Definition:** The number of pre-computations required for unsigned binary number is given by  $2^{w-1}$  and for signed binary number are given by  $2^{w-1} - 1$  where  $w$  is the size of window and is greater than 1 in a window method.

Let us take example of elliptical curve P-192, over prime field recommended by NIST to find the effect of window size on memory usages of WSN node.

#### Curve P-192

$p =$  6277101735386680763835789423207666416083908700390324961279

$r =$  6277101735386680763835789423176059013767194773182842284081

$s =$  3045ae6f c8422f64 ed579528 d38120ea e12196d5

$c =$  3099d2bb bfc2538 542dcd5f b078b6ef 5f3d6fe2 c745de65

$b =$  64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1

$G_x =$  188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012

$G_y =$  07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811

As seen the base point  $G(x, y)$  is having 192 bits each for  $x, y$  coordinates. All the values given in the above chart are in hexadecimal system. One hexadecimal corresponds to 4 bits of binary. If the window size  $w = 3$ , it will require 4 pre-computations for unsigned binary number and 3 for OCS method. These 4 points on curve P-192 will occupy  $4 * 192 * 2 = 1536$  bits of memory. In other words if 8 bits are equal to one byte, these 4 points will take 192 bytes of memory for storage for window size of 3. If the available memory is less than 192 bytes it will cause WSN node failure.



To avoid this problem, proposed Elastic window method will do stack depth analysis and will reduce size  $w$ , if the memory available is not sufficient. The detailed algorithm is given in the Figure 7.2.

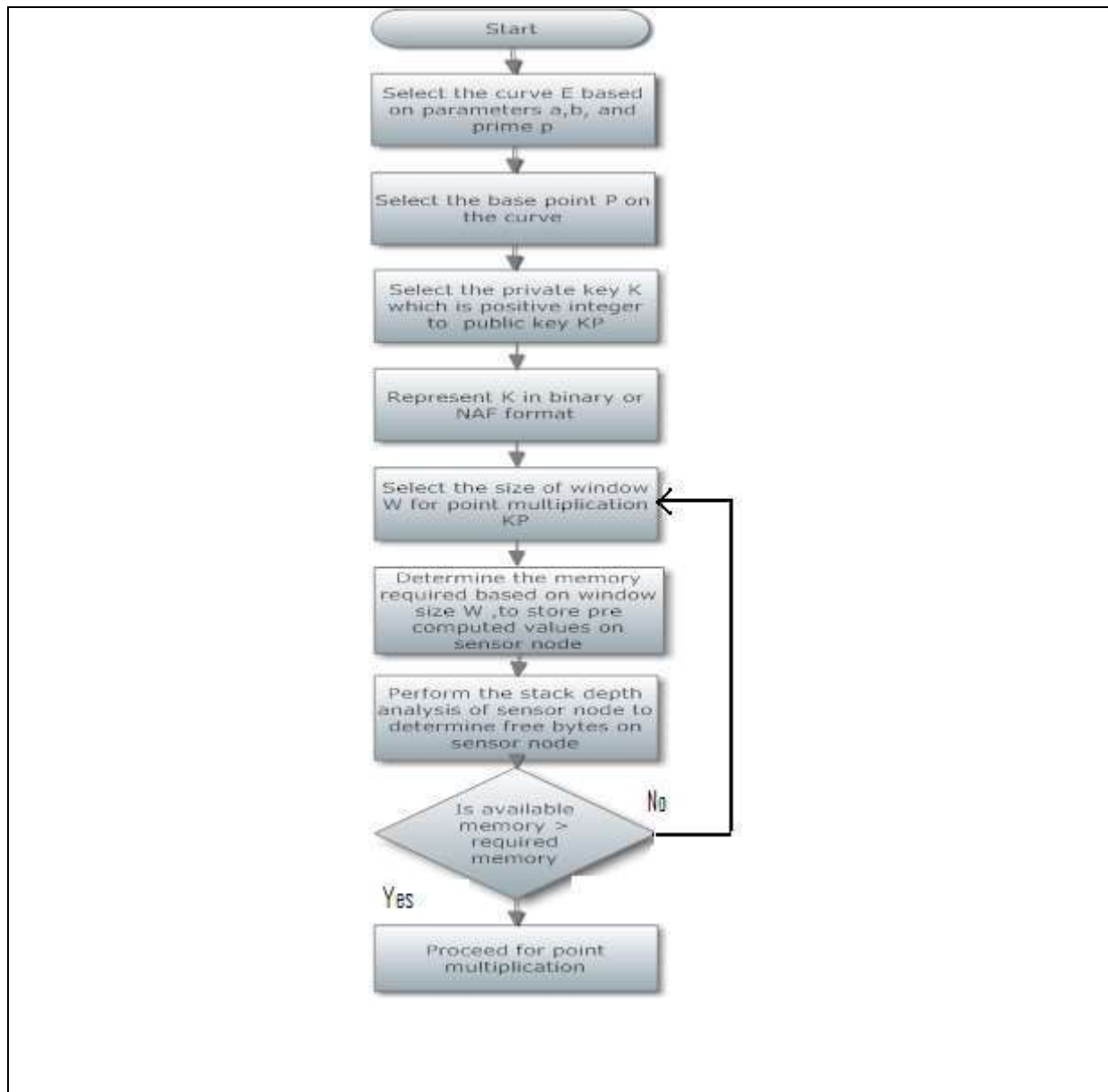


Figure 7. 2 Proposed Algorithm of Elastic Window Method for Scalar Multiplication on WSN Nodes

## **7.6 Summary**

Elastic window method for scalar multiplication on WSN platform can considerably remove the risk of node failure. The window size and available memory trade off can be done automatically with the new proposed algorithm. The proposed algorithm can be implemented on WSN platform with few simple instructions for e.g. if-else conditional statements and will not occupy much code size in ROM.

## Chapter 8 Algorithm Based on Hidden Generator Point Proposed for WSN to avoid man in the middle attack

---

### 8.1 Introduction

Today's software applications are mainly characterized by their component-based structures which are usually heterogeneous and distributed. Agent technology provides a method for handling increasing software complexity and supporting rapid and accurate decision making. A number of different approaches have emerged as candidates for the agent architecture, and at the same time, dozens of environments for modelling, testing and finally implementing agent-based systems have been developed. Software agent has been developed and it is a well-known MAS (multi-agent system) development kit supporting a world-wide agreed agent standard. In this chapter, MAS based on ECC for WSN has been presented. Recently, there is a trend for WSN that the *cluster head or group leader* rather than sensors will communicate to the base station which makes use of ECDH protocol. However, the algorithms related to this trend will inherently create chance for an attacker to make a “man-in-middle” attack, which is shown in Figure 8.1.

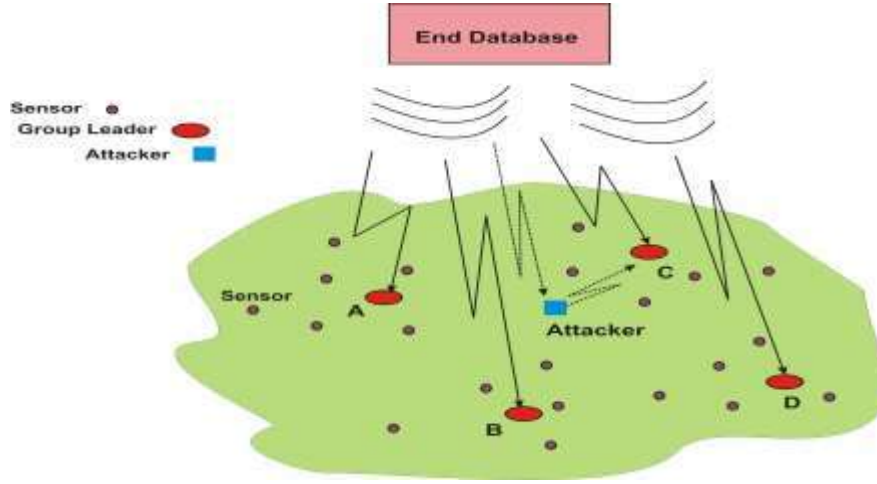


Figure 8. 1 Man-in-Middle Attack in WSN

## 8.2 Proposed New ECC Protocol Based On Hidden Generator Point

In order to express our new protocol of the hidden generator point we, without losing generality, we use an example. Let the Elliptic Curve has the following characteristics,

$$E : y^2 = x^3 + x + 1 \mod 23$$

$$p = 23, \quad a = 1, \quad b = 1$$

$$4a^3 + 27b^2 \mod 23 = 8 \neq 0.$$

The calculation results are shown below for the elliptic group  $E_p(a,b) = E_{23} = (1,1)$  which includes the point  $(4,0)$  corresponding to the single value  $y = 0$ .

The elliptic curve cryptography can be used to encrypt *plaintext messages*,  $M$ , into cipher texts. The plain text message  $M$  is encoded into a point from the finite set of points in the elliptic group,  $E_p(a,b)$ . First step consists in choosing a generator point,  $G \in E_p(a,b)$ , such

that the smallest value of  $n$  for which  $nG = \infty$  is a very large prime number. Normally the traditional ECC protocol is let the elliptic group  $E_p(a,b)$  and the generator point  $G$  be in public. The Alice and Bob selects their private keys, say  $n_A$  and  $n_B < n$  and compute the public keys  $P_A = n_A G$  and  $P_B = n_B G$ . Then, encrypt the message point  $P_M$  for the partner, say from Alice to Bob. So Alice (A) chose a random integer  $k_A$  and computes the cipher text pair of points  $P_C$  using Bob's public key  $P_B$ :

$$P_C = [(k_A G), (P_M + k_A P_B)]$$

Bob received the cipher text pair of points,  $P_C$  then multiplies the first point,  $(k_A G)$  with his private key,  $n_B$ , and then adds the result to the second point in the cipher text pair of points as shown below:

$$\begin{aligned} (P_M + k_A P_B) - [n_B (k_A G)] &= \\ &= (P_M + k_A P_B) - (n_B G k_A) \\ &= (P_M + k_A P_B) - (P_B k_A) \\ &= P_M \end{aligned}$$

which is the plain text point, corresponding to the plaintext message  $M$ . It is noted that only Bob can obtain retrieve the plan text information  $P_M$  by the private key  $n_B$ . The cryptographic strength of ECC lies in the difficulty for a cryptanalyst to determine the secret random number  $k$  from  $kP$  itself. The fast method to solve this problem is known as the elliptic curve logarithm problems (ECLP) as we have seen in Chapter 3 of the thesis.

It is clearly to see that ECC did not take care of the *man-in-the middle attacks* even ECC itself has its cryptographic strength as described above.

As above shown that the generator point  $G$  and elliptic group  $E_p(a,b)$  are in public. Now let's have a closer look at the elliptic group  $E_p(a,b)$ . In our above example, we pick the prime number  $p = 23$ ; (it is noted that this is only for explaining the new protocol, in real life

the  $p$  is bigger than this), we have quadratic residues group  $(p-1)/2 = 11$  and for this group the  $E_p(a,b)$  can be shown as below:

$$E_{23}(1,1) = \left\{ \begin{array}{cccccc} (0,1) & (0,22) & (1,7) & (1,16) & (3,10) & (3,13) & (4,0) \\ (5,4) & (5,19) & (6,4) & (6,19) & (7,11) & (7,12) & (9,7) \\ (9,16) & (11,3) & (11,20) & (12,4) & (12,19) & (13,7) & (13,16) \\ (17,3) & (17,20) & (18,3) & (18,20) & (19,5) & (19,18) & \end{array} \right\}$$

As we described in above that any point sitting in equation (3) can be appointed as generator point “ $G$ ,” in the traditional way (as in section II) the  $G$  is fixed and let it be in public. But now we are not going to do so. As the generator is hidden, there is no way to know which point is generator therefore the attacker cannot make the “*man-in-middle*” attack. Now we are going to show two ways to complete the ECC processing, namely,

- (1) Make protocol that has the common principle to work out the generator point, say from the distribution of elliptic  $E_p(a,b)$  group; or
- (2) by the new protocol to work out the  $P_M$  as shown below.

In order to make a common principle to work out the generator point  $G$  for Alice and Bob, say we are going to use the distribution of elliptic  $E_p(a,b)$  group, we need to check what it looks like. The distributions of  $E_{23}(1,1)$  is shown in Figure 8.2.

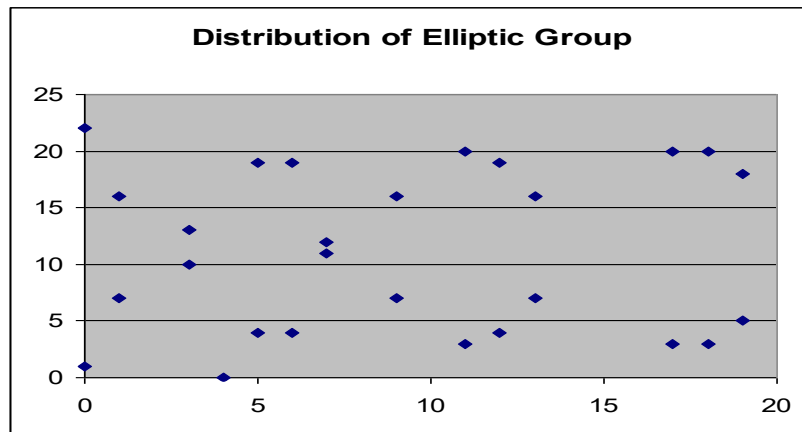


Figure 8. 2 Distribution of Elliptic Group E23 (1, 1).

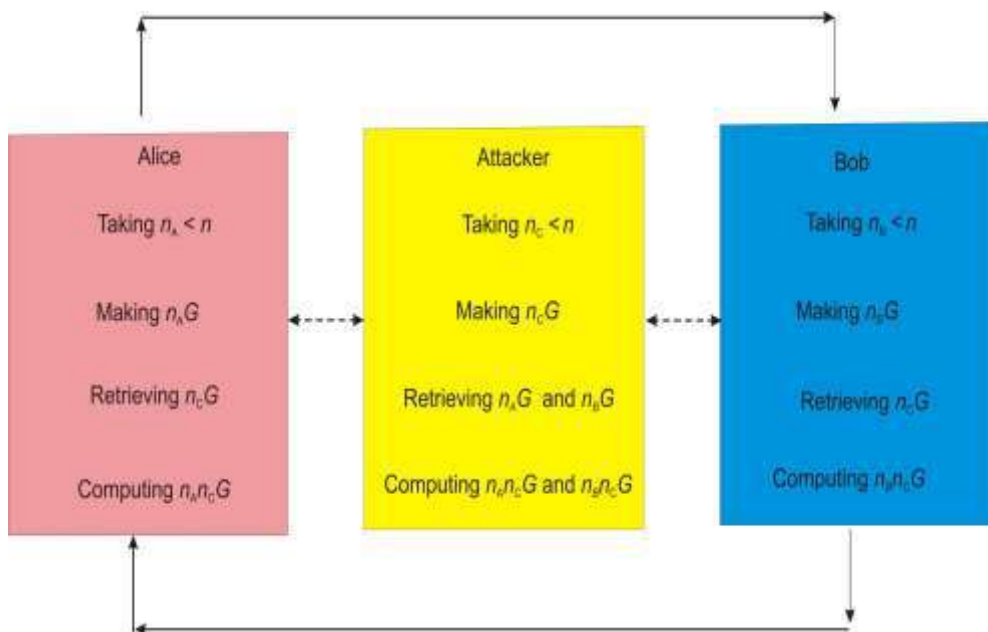


Figure 8. 3 A new protocol protecting the man-in-the-middle attack. As the  $G$  is not at the public, the attack cannot work out  $n_cG$  as traditional way does, so even the attacker can monitor the communication but no way to understand and attack the communications.



We now pick a generator point  $G$  by a character of the above distribution as shown in Figure 8.2, say we pick the  $G$  when  $G(a,b) \in E_p(a,b)$  with  $a = \max \{a\}, b = \max \{b\}$ . In this example,  $G = (19,18)$ . Note that we put  $a = \max \{a\}$  first, so choosing it  $a = 19$ , then choose  $b = \max \{b\}$ . The order is important, in this case it is not the  $G = (18,20)$ . When the generator point fixed, the protocol as shown in Figure 8.3 is executed which is resistant to man in the middle attack as point  $G$  is not public.

The other issue involved is the use of a trusted “certificate authority” (CA) in above protocol. In this case base station acts as certification authority for all the WSN nodes and sends a digitally signed “certificate” for every node. When any WSN node exchanges information with other WSN node it verifies it’s identity with a certificate send by the base station. This research proposes use of XOR operation to verify these digital certificates.

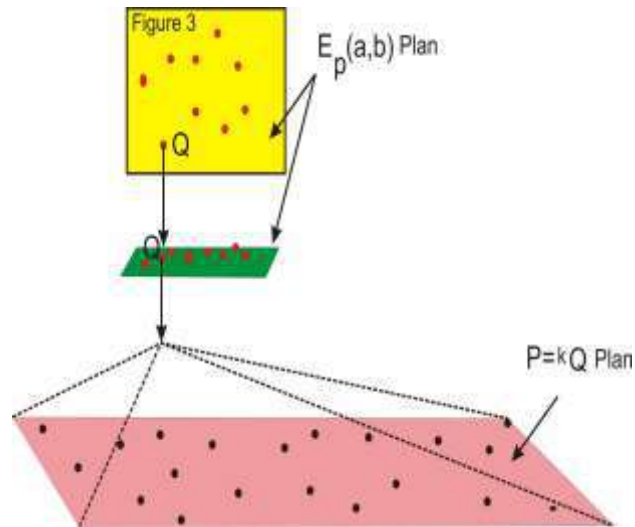


Figure 8. 4 Block diagram for hidden generator point principle. The top plan is figure 8.2. When a generator is fixed by the distribution of the elliptic group then the  $P = kQ$  plan formed.

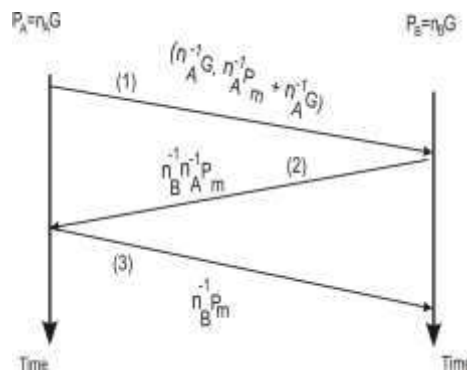


Figure 8. 5 A protocol for ECC with hidden generator point

Now let's turn to the second way, i.e., a new protocol to get the hidden generator point done. In the second way, we need to face the case that Alice has no information about Bob's public key as traditional way does. We may use, as an example, a protocol shown in Algorithm 8.1 to do this.

---

Algorithm 8.1 Proposed Protocol to avoid man in the middle attack

---

1. When Alice is going to send the message to Bob, Alice sends the pair of points  $P_C$

$$P_C = [(n_A^{-1}G), (n_A^{-1}P_M + n_A^{-1}G)]$$

Here,  $n_A^{-1}$  meets the equation:  $n_A^{-1}n_A = 1$ , we still called  $n_A^{-1}$  as private key for Alice .

But there is no need to worry about the public key as  $G$  is hidden at current situation.

So either  $P_A$  or  $P_B$  is not really useful in this case.

2. When Bob received  $P_C$ , he can operate as below:

$$n_A^{-1}P_M = n_A^{-1}P_M + n_A^{-1}G - n_A^{-1}G$$

3. Then, Bob can make  $P_D$  as below and sends it to Alice.

$$P_D = n_A^{-1}n_B^{-1}P_M$$

4. When Alice received  $P_D$ , Alice can make  $P_E$  and sent it to Bob.

$$P_E = (n_A)P_D = (n_A)(n_A^{-1}n_B^{-1}P_M) = n_B^{-1}P_M$$

5. Then when Bob received  $P_E$ , Bob can obtain the message related  $P_M$  that sent from Alice by

$$P_M = n_B n_B^{-1}P_M$$

---

As evident from above protocol , it makes excessive use of costly inversion operation. Here this protocol is presented for academic interest only. When the technology will get progressed

it may be useful on WSN platform. We can avoid inversion operations by use of projective coordinates as discussed in the Chapter 4 of the thesis .Use of projective coordinates and hidden generator point will provide very effective protection against SPA as well as man in the middle attacks for WSN nodes. The next section of this chapter provides ECC architecture based on multi agent system (MAS).

### 8.3 Multi-agent System Implementation of ECC Public Key

In the MAS architecture, every agent is attached to a gateway federate. We can have several agents in a group attached to the same gateway federate or a single agent attached to a separate gateway federate. At the same time, different federates can also be put in the same machine or different machines. In order to enable agents to interact with their environment, we develop three interfaces to play the role of interaction channels between the federate and the agent, namely, a *Rule Induction Agent*, which is handling the ECC cryptography rule as described in Algorithm 8.1, a *Dynamic Analysis Agent*, which is looking after related coding and decoding, and a *User Interface Agent*, which is dealing with interface processing. Both of them are transferred via the Object-to-Agent (O2A) communication channel provided by the agent toolkit. The framework of the MAS can be shown in Figure 8.6. Our work is concerned with the second area, namely to develop autonomous agents for representing entities in distributed simulations.

A MAS system comprised of multiple autonomous components needs to have certain characteristics ,each agent has incomplete capabilities to solve a problem;

- there is no global system control;
- data is decentralized; and
- computation is asynchronous.

That is, combining multiple agents in a framework presents a useful software engineering paradigm where problem-solving components are described as individual agents pursuing high-level goals.

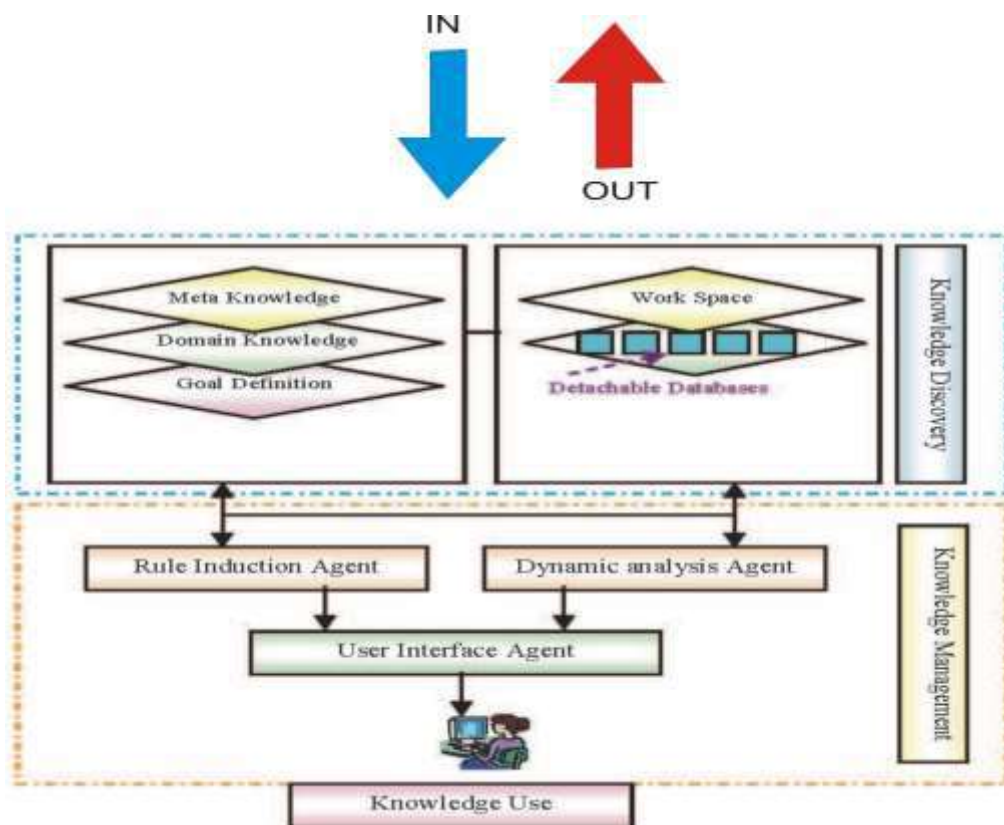


Figure 8. 6 Block Diagram of MAS Framework for ECC

It is noted that there are two ways, i.e. one is the date out and another is date in, so it is easily to communicate between two end systems, which is looking after by the agents. With multi-agent the whole system is more effective and efficiency for the ECC public key system.

Here we have used the MIRACL, one of the best multi-precision libraries. The benchmark program of MIRACL allows the user to quickly determine the time that will be required to implement any of the popular public key methods. For the test, we have put an example as demonstrates 1024 bit Diffie-Hellman, El Gamal and RSA and 168 bit Elliptic Curve Diffie-Hellman. With the same security level, the ECC has much smaller energy cost as the bit-number more than six times shorter. At above example the computing time cost is only 15% of the RSA with the same security level. The results obtained on MIRACL are given in the Appendix C of the thesis.

## **8.4 Summary**

As security issues in WSN are of prime importance, in recent years some cryptographic algorithms such as ECDH, ECDSA have obtained popularity due to properties that make them suitable for use in constrained environment such as WSN, where computing resources and power availability are limited. However, in the applications of ECC, in particular for the WSN, there are always so-called man-in-middle attacks. In this chapter we have presented two methods for protecting from man-in-middle attacks based on hidden generator point with ECC with multi agent system implementation. An effective and efficient system was designed and tested.

## Chapter 9 Proposed Uni-Coordinate System of ECC for WSN

---

### 9.1 Introduction

The keys in the ECC are very long having 160 to 512 bits depending on the required security level. In most of the standard protocols like ECDH, ECDSA, WSN nodes share the key before actual information exchange starts. Due to exchange of long keys sensor nodes get strain on already limited packet size, bandwidth and energy resources. To avoid this problem, this chapter introduces a short note in which there is no need to transfer both coordinates of the key. In this proposed scheme WSN node transfers only one coordinate and one extra bit. The receiving node calculates the other coordinate by solving the quadratic equation of the elliptical curve. The next subsections give mathematical proof and analysis for the same.

### 9.2 Quadratic equations in field of odd characteristics:

Solving quadratic equation is an important operation in the elliptic curves, where it is used to obtain the y-co-ordinate of a point given its x-co-ordinate. Assume we wish to solve the equation  $x^2 \equiv a \pmod{p}$ . To see whether such an equation actually has a solution, the Legendre Symbol  $(a/p)$  which is equal to 1 if  $a$  is a square modulo  $p$ , 0 if  $a \equiv 0 \pmod{p}$  or  $-1$  otherwise is used. To compute the Legendre symbol the following method based on quadratic reciprocity [29] can be used.

---

Algorithm 9.1 Legendre Symbol

---

*Input :  $a$  and  $p$ .*

*Output :  $(\frac{a}{p}) \in \{1, 0, -1\}$ .*

1. *If  $a \equiv 0 \pmod{p}$  then return 0.*

2.  *$x \leftarrow a, y \leftarrow p, L \leftarrow 1$ .*

3.  *$x \leftarrow x \pmod{y}$ .*

4. *If  $x > y/2$  then do :*

5.  *$x \leftarrow y - x$ ,*

6. *If  $y \equiv 3 \pmod{4}$  then  $L \leftarrow -L$ .*

7. *While  $x \equiv 0 \pmod{4}$  do  $x \leftarrow x/4$ .*

8. *If  $x \equiv 0 \pmod{2}$  then do :*

9.  *$x \leftarrow x/2$ ,*

10. *If  $y \equiv \pm 3 \pmod{8}$  then  $L \leftarrow -L$ .*

11. *If  $x = 1$  then return  $L$ .*

12. *If  $x \equiv 3 \pmod{4}$  and  $y \equiv 3 \pmod{4}$  then  $L \leftarrow -L$ .*

13. *Swap  $x$  and  $y$  and go to step 3.*

---

Alternatively we could compute  $a^{(p-1)/2} \pmod{p}$ . It can thus be decided whether  $a$  is or not square. If  $a \equiv 0 \pmod{p}$  then  $a$  has only one square root modulo  $p$  which is 0. If  $(a/p) = 1$  then there are two square roots modulo  $p$  and we need to determine one of them. The following algorithm is based on a method of Tonelli and shanks [29].



#### Algorithm 9. 2 Square root modulo p

---

#### Algorithm 9.2 Square root modulo $p$

---

*Input:*  $a$  and  $p$  such that  $(\frac{a}{p}) = 1$ .

*Output:*  $x$  such that  $x^2 \equiv a \pmod{p}$ .

1. Choose random  $n$  until one is found such that  $(\frac{n}{p}) = -1$ .
  2. Let  $e, q$  be integers such that  $q$  is odd and  $p - 1 = 2^e q$ .
  3.  $y \leftarrow n^q \pmod{p}, r \leftarrow e, x \leftarrow a^{(q-1)/2} \pmod{p}$ ,
  4.  $b \leftarrow ax^2 \pmod{p}, x \leftarrow ax \pmod{p}$ .
  5. While  $b \not\equiv 1 \pmod{p}$  do:
  6. Find the smallest  $m$  such that  $b^{2^m} \equiv 1 \pmod{p}$ ,
  7.  $t \leftarrow y^{2^{r-m-1}} \pmod{p}, y \leftarrow t^2 \pmod{p}, r \leftarrow m$ ,
  8.  $x \leftarrow xt \pmod{p}, b \leftarrow by \pmod{p}$ .
  9. Return  $x$ .
- 

### 9.3 Solving quadratic equations in binary field.

As per [29], an equation of the form  $x^2 + \beta = 0$  is trivially solved in  $F_{2^n}$  by writing its (double) root  $x_0$  explicitly as  $x_0 = \beta^{2^{n-1}}$ . Other non-trivial quadratic equations can always be brought to the canonical form

$$x^2 + x + \beta = 0$$

This equation has solutions in  $F_{2^n}$  if and only if  $\text{Tr}_{q/2}(\beta) = 0$ . If  $x_0$  is such a solution, then so is  $x_0 + 1$ .

The Procedure for finding a solution varies according to the parity of  $n$ . If  $n$  is odd, an explicit solution is given by  $x_0 = T(\beta)$ , where  $T$ , the half – track function, is defined by

$$T(\beta) = \sum_{j=0}^{(n-1)/2} \beta^{2^{2j}}$$

It can be verified by direct inspection that  $T(\beta)^2 + T(\beta) = \beta + \text{Tr}_{q/2}(\beta)$ , which verifies the solution when  $\text{Tr}_{q/2}(\beta) = 0$ .

When  $n$  is even, the half-trace will not do, and a solution is found using the following procedure, Let  $\delta \in F_{2^n}$  be such that  $\text{Tr}_{q/2}(\delta) = 1$ . Such an element can be obtained either by randomly drawing field elements until one of the right trace is found (with a probability of one half in each try), or by deterministically computing the traces of the basis elements  $\alpha_0, \alpha_1, \dots, \alpha_{n-1}$ . At least one basis element must have trace one. In practice, computing these basis traces can be a good investment, as the vector

$$t = (\text{Tr}_{q/2}(\alpha_0), \text{Tr}_{q/2}(\alpha_1), \dots, \text{Tr}_{q/2}(\alpha_{n-1}))$$

is useful for computing traces of arbitrary field elements. With  $\delta$  at hand, & solution  $x_0$  to Equation (II.6) is given by

$$x_0 = \sum_{i=0}^{n-2} \left( \sum_{j=i+1}^{n-1} \delta^{2j} \right) \beta^{2i}$$

To verify that  $x_0$  is indeed a solution, we compute

$$\begin{aligned} x_0^2 + x_0 &= \sum_{i=1}^{n-1} \left( \sum_{j=i+1}^n \delta^{2j} \right) \beta^{2i} + \sum_{i=0}^{n-2} \left( \sum_{j=i+1}^{n-1} \delta^{2j} \right) \beta^{2i} \\ &= \delta(\beta^{2n-1} + \beta^{2n-2} + \dots + \beta^2) + (\delta^{2n-1} + \delta^{2n-2} + \dots + \delta^2)\beta \\ &= \delta \text{Tr}_{q^2}(\beta) + \beta, \end{aligned}$$

where the last equality follows from  $\delta^{2n-1} + \delta^{2n-2} + \dots + \delta^2 = \text{Tr}_{q^2}(\delta) + \delta = 1 + \delta$

Thus,  $x_0^2 + x_0 = \beta$  if and only if  $\text{Tr}_{q^2}(\beta) = 0$ , as desired.

## 9.4 Summary

The use of uni-coordinate system will save packet size, bandwidth and energy resources of sensor node considerably as overheads get reduced by almost 50%.

## Chapter 10 Summary of the Research

---

Elliptical Curve Cryptography is suitable for current generation WSN nodes like Imote2 with the careful designing of modular reduction, modular inversions algorithms. The results obtained on MIRACL are encouraging in this regard.

While choosing the finite field arithmetic proper care has to be taken to choose the base, options available are prime, binary and special field bases.

There are many choices of coordinate's representations and these have a significant impact on the computational cost of cryptographic protocols like ECDH, ECDSA. Based on various results obtained on MIRACL, this thesis recommends *AJM* coordinate system with one input in *affine coordinate* and other in *Jacobian coordinate* with output in *Modified Jacobian coordinate* for point addition algorithm. This coordinate system will offer best results if the constant  $a = -3$  chosen. (All curves used in MIRACL are having value of  $a = -3$  for this reason. The list of the NIST curves is given in the Appendix A of the thesis). One of the input is chosen in *affine coordinate* because of the faster implementations available and because fewer variables were required in this case.

For point doubling algorithm, this thesis recommends *Modified Jacobian MM* or *Mixed Jacobian MJ* coordinate system. Experiments carried out on MIRACL crypto library clearly indicate that modular inversion is the costly operation for WSN and it has to be avoided in any case.

*AJM*, *MM* and *MJ* coordinate system do not use any modular inversion operation. Also conversions among three *Jacobian variants* are most efficient. All these circumstances leads to better performance on WSN node while implementing the ECDH and ECDSA protocols.

Other method for attempting to speed up the EC computations on WSN platform is to transform the base point of curve from *affine coordinate* to *projective coordinate*. The use of *projective coordinate* will eliminate the use of modular inversion operation. But in this case, one modular inversion will be needed at the end to convert final result back to *affine coordinate* system.

Doing *two in one scalar multiplication* ( $k_1P + k_2Q$ ) process which is always referred in literature as Shamir's trick [25] will improve the signature verification time for ECDSA algorithm. The results obtained for Shamir's trick on MIRACL library are given in the Appendix B of the thesis for verification of signature process.

The projective coordinate system has shown better results as compared to affine coordinate system due to absence of inversion operation in the former one.

To reduce the timings of scalar multiplication, recoding of the integer has done with new and simple innovative OCS method by using one's complement subtraction property and has shown good results as compared with NAF method. As the new method requires only bitwise subtraction, it puts very little load on microprocessor of sensor to get this recoding done as compared to NAF method. This new method will also provide resistance to simple power analysis attack if combined with window method for scalar multiplication.

The WSN node consists of limited memory for storage so to avoid SPA attack is very difficult in such environment. Our proposed window OCS method based on windowing principle will execute point addition and doubling in uniform sequence irrespective of scalar and will make attacker difficult to detect the secret scalar by using SPA. The window size  $w$  is a matter of trade off between the available memory and performance.

The optimum window size  $w$  for sliding scalar multiplication can be selected with our innovative Elastic window method which will considerably remove the risk of WSN node

failure. The window size and available memory trade off can be done automatically with the Elastic Window algorithm. The proposed algorithm can be implemented with very few instructions and will not occupy much code size in ROM.

ECC has been paid more attentions in recent years due to properties that make them suitable for use in constrained environment such as WSN, where computing resources and power availability are limited. However, if the authentications of the node are not done properly there is a possibility of man-in-middle attacks. This research has presented two innovative methods for protecting from man-in-middle attacks based on hidden generator point concept with MAS implementation. At first instance this system looks computationally intensive, but as the technology will progress it will show its merits. Also by making use of projective coordinates these inversions can be avoided.

The last part of thesis has shown how to use a single coordinate of public key to save packet size and bandwidth of the network and has given mathematical modeling for the same. The use of uni-coordinate system will save 50 % of packet size, bandwidth and energy resources of sensor node during encryption and decryption process.

## Chapter 11 Future Scope

---

The *cloud computing* will play major role in the implementation of ECC on WSN for achieving the overall security. In this case base station will do authentication tasks of the participating nodes .The computational intensive task will be handled by *cloud* and sensor nodes will get the ready made keys for the information exchange. Due to this shifting of the responsibilities, it is possible for the sensor nodes to use all available resources for application purpose the reason they have deployed in the field.

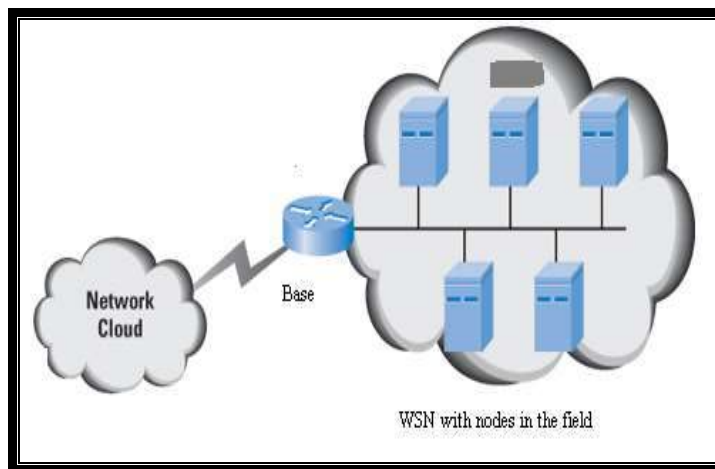


Figure 11. 1 Cloud Computing Architecture for WSN

## Bibliography

---

- [1] I. F. Akyildiz, and I. H. Kasimoglu, "Wireless sensor and actor networks: research challenges," *Ad Hoc Networks*, vol. 2, no. 4, pp. 351-367, 2004.
- [2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam *et al.*, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393-422, 2002.
- [3] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Computer Networks*, vol. 47, no. 4, pp. 445-487, 2005.
- [4] Y. Hitchcock, P. Montague, G. Carter *et al.*, "The Security of Fixed versus Random Elliptic Curves in Cryptography," *Information Security and Privacy*, Lecture Notes in Computer Science, pp. 220-220: Springer Berlin / Heidelberg, 2003.
- [5] Z. L. John Paul Walters, and a. V. C. Weisong Shi, "Wireless Sensor Network Security: A Survey," *Security in Distributed, Grid, and Pervasive Computing*, Yang Xiao, ed.: Auerbach Publications, CRC Press, 2006.
- [6] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems " *Communication ACM*, vol. 21, no. 2, pp. 120-126, 1978.
- [7] N.Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203-209, 1987.



- [8] G. Gunnar, K. Jens-Peter, and S. Berk, *Public Key Cryptography in Sensor Networks—Revisited*, 2005.
- [9] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks " *Communication ACM* vol. 47, no. 6, pp. 53-57, 2004.
- [10] A. Liu, and P. Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks," in Proceedings of the 7th international conference on Information processing in sensor networks, 2008.
- [11] R. Watro, D. Kong, S.-f. Cuti *et al.*, "TinyPK: securing sensor networks with public key technology," in Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks Washington DC, USA 2004.
- [12] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography." pp. 71-80.
- [13] N. Gura, A. Patel, and A. Wander, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," in Proceedings of the 2004 Workshop on Cryptographic Hardware and Embedded Systems (CHES) August 2004.
- [14] NIST. "<http://csrc.nist.gov/CryptoToolkit/dss/ecdsa/NISTReCur.pdf>."
- [15] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography* 1999.
- [16] S. C. Shantz, *From euclid's gcd to montgomery multiplication to the great divide*, Technical Report, Sun Microsystems Laboratories TR -2001-95 2001.

- [17] A. D. Woodbury, D. V. Bailey, and C. Paar, "Elliptic curve cryptography on smart cards without coprocessors. ," Bristol, UK, September 2000.
- [18] Author ed.^eds., "Efficient elliptic curve exponentiation," *ICICS'97, volume 1334 of LNCS, pages 282--290*: Springer--Verlag, 1997, p.^pp. Pages.
- [19] J. Lopez, and R. Dahab., *An overview of elliptic curve cryptography*, Technical report ,Institute of Computing, Sate University of Campinas, Sao Paulo, Brazil,, May 2000.
- [20] D. Hankerson, J. L. Hernandez, and A. Menezes, "Software Implementation of Elliptic Curve Cryptography over Binary Fields, CHES," 2000.
- [21] <http://www.xbow.com/Products/productdetails.aspx?sid=253>
- [22] W. Diffie, and M. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on*, vol. 22, no. 6, pp. 644-654, 1976.
- [23] American Bankers Association. *ANSI X9.62-1998: Public KeyCryptography for the Financial Services Industry: the EllipticCurve Digital Signature Algorithm (ECDSA)*, 1999.
- [24] H. Cohen, and G. Frey, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*: Chapman and Hall/CRC, 2006.
- [25] T. ElGamal, "A public key cryptosystem and a sig-nature scheme based on discrete logarithms," IEEE Transactions on Information Theory, vol. 31, no.4,pp. 469- 472, 1985.
- [26] E. Trichina, and A. Bellezza, "Implementation of Elliptic Curve Cryptography with Built-In Counter Measures against Side Channel Attacks," *Cryptographic Hardware and Embedded Systems - CHES 2002*, Lecture Notes in Computer Science, pp. 297-312: Springer Berlin / Heidelberg, 2003.

- [27] Bodo Möller, Securing Elliptic Curve Point Multiplication against Side-Channel Attacks, Information Security – ISC 2001, LNCS 2200. Springer-Verlag, 2001. 324–334. ISBN 3-540-42662-0, ISSN 0302-9743.
- [28] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” IEEE Transactions on Information Theory, vol. 31, no.4, pp. 469–472, 1985.
- [29] Blake, I. F., Seroussi, G., and Smart, N. P. Elliptic Curves in Cryptography, vol. 265 of London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.
- [30] N. Thi’erault, SPA resistant left-to-right integer recodings, Selected Areas in Cryptography – SAC 2005 (B. Preneel and S.E. Tavares, eds.), Lecture Notes in Computer Science, vol. 3897, Springer-Verlag, 2006, pp. 345–358..”
- [31] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS’03)*, pages 52–61, October 2003.
- [32] D. Liu and P. Ning. Improving key pre-distribution with deployment knowledge in static sensor networks. *ACM Transactions on Sensor Networks*, 1(2):204–239, November 2005.
- [33] D. Liu and P. Ning. Multi-level mTESLA: Broadcast authentication for distributed sensor networks. *ACM Transactions in Embedded Computing Systems (TECS)*, 3(4):800–836, 2004.

- [34] D. Liu, P. Ning, S. Zhu, and S. Jajodia. Practical broadcast authentication in sensor networks. In *Proceedings of the 2<sup>nd</sup> Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2005)*, July 2005.
- [35] H. Wang and Q. Li. Efficient implementation of public key cryptosystems on mote sensors. In *Proceedings of International Conference on Information and Communication Security (ICICS)*, pages 519–528, Dec. 2006..”
- [36] A. Liu, P. Kampanakis, and P. Ning. TinyECC: Elliptic curve cryptography for sensor networks (version 0.3). <http://discovery.csc.ncsu.edu/software/TinyECC/>.
- [37] D. Malan, M. Welsh, and M. Smith. A public-key infrastructure for key distribution in tinys based on elliptic curve cryptography. In *Proceedings of IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, pages 71–80, 2004.
- [38] Certicom Research. Standards for efficient cryptography SEC 1: Elliptic curve cryptography. [http://www.secg.org/download/aid-385/sec1\\_final.pdf](http://www.secg.org/download/aid-385/sec1_final.pdf), September 2000.
- [39] N. Gura, A. Patel, and A. Wander. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *Proceedings of the 2004 Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004)*, pages 119–132, August 2004.
- [40] An Liu, Peng Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks," in *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008)*, SPOTS Track, pages 245--256, April 2008 .
- [41] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, “Wireless sensor networks: a survey,”, *Computer Networks(Elsevier) Journal*, vol. 38, no. 4, pp. 393-422, March 2002.

- [42] D. Steere, A. Baptista, D. McNamee, C. Pu, J. Walpole, "Research challenges in environmental observation and forecasting systems," in *Proc. ACM/IEEE MOBICOM '00*, Boston, August 2000.
- [43] L. Schwiebert, S. K. S. Gupta, and J. Weinmann, "Research challenges in wireless networks of biomedical sensors," in *Proc. ACM/IEEE MOBICOM '01*, pp. 151 -165, 2001.
- [44] S.D. Feller, et al., "Tracking and imaging humans on heterogeneous infrared sensor arrays for law enforcement applications," *SPIE Aerosense 2002*, April, 2002..
- [45] S.D. Feller, et al., "Tracking and imaging humans on heterogeneous infrared sensor array for tactical applications," *SPIE Aerosense 2002*, April, 2002..
- [46] MICA Motes and Sensors, available at <http://www.xbow.com/Products/WirelessSensorNetworks.htm>.."
- [47] [http://www.powercomgroup.com/uploads/1/Fleck\\_tech\\_preview.pdf](http://www.powercomgroup.com/uploads/1/Fleck_tech_preview.pdf).
- [48] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proc. IEEE/ACM Information Processing in Sensor Networks (IPSN) - Track on Platforms, Tools and Design Methods for Networked Embedded Systems (SPOTS)*, 2005.
- [49] Shih, E. et al., Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks, *ACM Mobicom '01*, pp. 272-286, Rome, Italy, July 2001.

- [50] A. Sinha and A. Chandrakasan, "Dynamic Power Management in Wireless Sensor Networks," *IEEE Design and Test of Computers*, March/April 2001.
- [51] <http://www.filesland.com/companies/Naval-Academy/SNetSim.html>.
- [52] S. Hedetniemi and A. Liestman, A Survey of Gossiping and Broadcasting in Communication Networks, *Networks*, Vol. 18, No. 4, pp. 319-349, 1988.
- [53] A. D. Wood, and J. A. Stankovic, "Denial of service in sensor networks," *Computer*, vol. 35, no. 10, pp. 54-62, 2002.
- [54] Raymond D.R. Midkiff.S.F, "Denial of Service in Wireless Sensor Network: Attacks and Defenses", *IEEE Pervasive Computing*, Vol:7, Issue 1, PP: 74 - 81, March 2008.
- [55] S Mohit, "Security In Wireless Sensor Networks - A Layer Based Classification", *Cerias Tech Report 2007-04*.
- [56] P. Kocher, J. Jaffe and B. Jun, "Differential Power Analysis", *CRYPTO 1999*, Santa Barbara, California, USA, LNCS 1666, pages 388–397, August 1999.
- [57] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*: Springer, 2004.
- [58] J.S. Coron, "Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems", *CHES 1999*, Worcester, MA, USA, LNCS 1717, pages 292–302, Springer, August 1999.
- [59] T.Kavitha, and D.Sridharan, "Security Vulnerabilities In Wireless Sensor Networks: A Survey " *Journal of Information Assurance and Security 5 (2010) 031-044*, pp. 31-44, 2010.
- [60] J. Newsome, E. Shi, D. Song *et al.*, "The sybil attack in sensor networks: analysis \& defenses " in *Proceedings of the third international symposium on Information processing in sensor networks* Berkeley, California, USA 2004.

- [61] Jianmin Zhang, Qingmin Cui, Xiande Liu, "An Efficient Key Management Scheme for Wireless Sensor Networks in Hostile Environments", International Conference on Multimedia Information Networking and Security, 2009 .
- [62] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks " in Proceedings of the 2nd international conference on Embedded networked sensor systems Baltimore, MD, USA 2004.
- [63] A. Perrig, R. Szewczyk, J. D. Tygar *et al.*, "SPINS: security protocols for sensor networks " *Wireless Networks*, vol. 8, no. 5, pp. 521-534 2002.
- [64] Author ed.^eds., "Wireless Sensor Networks :A system perspective," Artech House London, 2005, p.^pp. Pages.
- [65] V. S. Miller, "Use of Elliptic Curves in Cryptography," *Advances in Cryptology - CRYPTO '85: Proceedings*, pp. 417-426: Springer-Verlag, 1986.
- [66] Taher ElGamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", IEEE Transactions on Information Theory, v. IT-31, n. 4, 1985, pp469–472 .
- [67] IEEE Std 1363-2000: Specifications for Public Key Cryptography. August 2000 .
- [68] ANSI X9.62, "American National Standard for Financial Services - Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry - The Elliptic Curve Digital Signature Algorithm (ECDSA)", draft, ASC X9 Secretariat - American Bankers Association, December 1997.

- [69] ANSI X9.30-1, "American National Standard for Financial Services - Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry - Part 1: The Digital Signature Algorithm (DSA)", ASC X9 Secretariat - American Bankers Association, 1995.
- [70] <http://www.nist.gov/index.html>.
- [71] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*: CRC press, 1996, 1996.
- [72] P. L. Montgomery, "Modular multiplication without trial division " *Math. Computation*, (1985), 44:519-521.
- [73] A. Karatsuba, and Ofman, " *Multiplication of multidigit numbers by automata*" *Soviet Physics-Doklady* 7 pp. 595-596, 1963.
- [74] Donald E. Knuth, "The Art of Computer Programming," 2. *Seminumerical Algorithms Addison-Wesley*, 1981.
- [75] C. H. Gebotys, *Security in Embedded Devices*: Springer, 2010.
- [76] Author ed.^eds., "Efficient elliptic curve exponentiation using mixed coordinates," *In K. Ohta and D. Pei, editors, Advances in Cryptology - ASIACRYPT'98, volume 1514 of LNCS, pages 51--65. Springer--Verlag 1998.*, p.^pp. Pages.
- [77] Henri Cohen, Atsuko Miyaji, and Takatoshi Ono. Efficient elliptic curve exponentiation using mixed coordinates. In ASIACRYPT '98: Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security, pages 51–65, London, UK, 1998. Springer-Verlag.
- [78] S. Balasubramanian, and D. Aksoy, "Adaptive energy-efficient registration and online scheduling for asymmetric wireless sensor networks," *Computer Networks*, vol. In Press, Uncorrected Proof, pp. 1168.



- [79] [http://www.atmel.com/dyn/products/product\\_card.asp?PN=ATmega128A](http://www.atmel.com/dyn/products/product_card.asp?PN=ATmega128A).
- [80] Donald E. Knuth. Seminumerical Algorithms, volume 2 of The Art of Computer Programming. Addison-Wesley, Reading, Massachusetts, second edition, 1981.
- [81] Standard Specifications for Public Key Cryptography, IEEE Standard 1363, 2000.
- [82] A.D. Booth, A signed binary multiplication technique, Journal of applied Mathematics 4 (2)(1951)236-240.
- [83] G. W. Reitwiesner, "Binary arithmetic," Advances in Computers, vol. 1, pp. 231–308, 1960.
- [84] F. Morain and J. Olivos, *Speeding up the computations on an elliptic curve using addition-subtraction chains*. RAIRO Inform. Theory **24** (1990), 531–543.
- [85] S. Arno and F.S. Wheeler, *Signed digit representations of minimal Hamming weight*. IEEE Transactions on Computers **42** (1993), 1007–1010.
- [86] K. Okeya, Signed binary representations revisited, Proceedings of CRYPTO'04 (2004) 123–139.
- [87] A. C. Gillie, *Binary Arithmetic and Boolean Algebra*: Mc Graw-Hill Inc New York, 1965.
- [88] Angelo C Gillie, "Binary Arithmetic and Boolean algebra," McGRAW-HILL Book Company, 1965. pp53.

- [89] P. Kocher, J. Jaffe, B. Jun, "Differential Power Analysis," technical report, 1998; later published in *Advances in Cryptology - Crypto 99 Proceedings, Lecture Notes In Computer Science Vol. 1666*, M. Wiener, ed., Springer-Verlag, 1999.
- [90] T. Messerges, E. Dabbish, and R. Sloan, "Investigations of Power Analysis Attacks on Smartcards," *Proc. USENIX Workshop Smartcard Technology (Smartcard '99)*, pp. 151-161, 1999.
- [91] T. Messerges, E. Dabbish, and R. Sloan, "Investigations of Power Analysis Attacks on Smartcards," *IEEE Trans. Computers*, vol. 51, no. 5, May 2002.
- [92] Elisabeth Oswald: Enhancing Simple Power-Analysis Attacks on Elliptic Curve Cryptosystems. *CHES 2002*: 82-97.
- [93] Christophe Clavier and Marc Joye. Universal exponentiation algorithm a first step towards provable SPA-resistance. In *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 300-308. Springer, 2001.

## Appendix A NIST Recommended Curves for Cryptography

---

This appendix presents examples of elliptic curves whose groups of rational points contain large prime subgroups. Section 1 shows curves over finite fields  $F_q$ , with  $q = p$ , a large prime, while Section 2 shows curves over  $F_q$ , with  $q = 2^n$ . Unless explicitly noted otherwise, the curves are ‘random’, in the sense that their relevant coefficients were drawn at random, with uniform probability, and the orders of their groups of rational points were determined using the point counting algorithms. In each case, a number of random curves  $E$  were generated, and the order of the group  $E(F_q)$  determined, until a satisfactory one was found.

### Section 1. Odd Characteristic

The examples in this section describe curves over fields  $F_p$ , where  $p$  is a large prime. The curve equations are of the form

$$E: Y^2 = X^3 + aX + b, \quad a, b \in F_p.$$

For each curve, the values of  $p$ ,  $a$ ,  $b$ , and  $\#E(F_p)$  are listed, with elements of  $F_p$  shown as integers in the range  $\{0, 1, \dots, p-1\}$ , in decimal notation. When  $\#E(F_p)$  is composite, it is also shown factored as  $s \cdot r$ , where  $s$  is a small positive integer, and  $r$  is prime. Large integers might be broken into multiple lines, with a backslash at the end of a line indicating that the number is continued in the next line.

Examples 1-7 show ‘random’ curves, as described above. In these examples, the values of  $p$  are all of the form  $2^k + c$ ,  $c$  a small positive integer, so the ‘size’ of a field element is self-evident. The curves in examples 8-11 were generated with the CM method. For these examples, the value  $[\log_2 p]$  is shown (since  $p$  has no special form), as are the discriminant  $-D$  and the class number  $h_D$ .

In all cases, the curve initially obtained was renormalized with a transformation of the form  $a \rightarrow u^4 a, b \rightarrow u^6 b, u \neq 0$ , to make coefficient a small integer. As discussed in Chapter III, the resulting curve is isomorphic to the original one.

EXAMPLE 1.  $p = 2^{130} + 169$

$$= 1361129467683753853853498429727072845993,$$

$$a = 3,$$

$$b = 1043498151013573141076033119958062900890,$$

$$\#E(\mathbb{F}_p) = 1361129467683753853808807784495688874237$$

( $a$  a prime number).

EXAMPLE 2.  $p = 2^{130} + 169$

$$= 1361129467683753853853498429727072845993,$$

$$a = 1,$$

$$b = 1230929586093851880935564157041535079194,$$

$$\# E(F_p) = 1361129467683753853846060531160085896483$$

( a prime number).

$$\text{EXAMPLE 3. } p = 2^{160} + 7$$

$$= 146150163733090291820368432716283019655932542983,$$

$$a = 10,$$

$$b = 1343632762150092499701637438970764818528075565078,$$

$$\# E(F_p) = 1461501637330902918203683518218126812711137002561$$

( a prime number).

$$\text{EXAMPLE 4. } p = 2^{160} + 7$$

$$= 146150163733090291820368432716283019655932542983,$$

$$a = 1,$$

$$b = 1461501637330902918203683038630093524408650319587,$$

$$\# E(F_p) = 1461501637330902918203683038630093524408650319587$$

( a prime number).

EXAMPLE 5.  $p = 2^{190} + 129$

$$= 15692754338466701909589473558019166040255888611160 \backslash$$

$$a = 10,$$

$$b = 13484624114143613126110541131169310875806949186774 \backslash$$

$$22294274,$$

$$\# E(F_p) = 15692754338466701909589473557802870403052555408969 \backslash$$

$$46997883,$$

( a prime number).

EXAMPLE 6.  $p = 2^{190} + 129$

$$= 15692754338466701909589473558019166040255888611160 \backslash$$

$$a = 1,$$

$$b = 12352246712371885871866833148430395515491455516523 \backslash$$

$$48919785,$$

$$\# E(F_p) = 15692754338466701909589473557448604281873393792782 \backslash$$

34198947,

( a prime number).

EXAMPLE 7.  $p = 2^{230} + 67$

=17254365866976409468586889655692563631127772430425 \

96638790631055949891

$a = 7,$

$b = 30760627165932116708009308342886016941744188615122 \setminus$

817540619633362515,

$\# E(F_p) = 17254365866976409468586889655692563495678763846462 \setminus$

09701190542123355279,

$= 3.57514552889921364895289632185641878318929212821 \setminus$

5403233730180707785093.

EXAMPLE 8. CM method:  $D = 120$ , class number  $h_D = 4$ .

$$p = 65455032684134289174663529622762084395449925683109 \setminus$$

$$5265159541652712020456264658138887199159,$$

$$[\log_2 p] = 299,$$

$$a = 1,$$

$$b = 36144049423228325504814614498592213402384746101106 \setminus$$

$$1544601950323468977393822343563902480881,$$

$$\# E(F_p) = 65455032684134289174663529622762084395449925575241 \setminus$$

$$08587449568038781938244528858339260141966,$$

$$= 2.32727516342067144587331764811381042197724962787 \setminus$$

$$620542937247840193909691222644291630070983.$$

EXAMPLE 9. CM method:  $D = 532$ , class number  $h_D = 4$ .

$$p = 7535163018303183031237089471027567747356575330769048527 \setminus$$

$$5188329021453480578403168297870032548617,$$

$$[\log_2 p] = 299,$$

$$a = 5,$$



$$b = 88724011078561722486617538567703959028485726225214 \setminus$$

$$9486851149501578553921103683825690162,$$

$$\# E(\mathbb{F}_p) = 75351630183031237089471027567747356575330769209158 \setminus$$

$$0197738211634229731628612489372399302238,$$

$$= 2.37675815091515618544735513783873678287665384604 \setminus$$

$$5790098869105817114865814306244686199651119.$$

EXAMPLE 10. CM method:  $D = 120$ , class number  $h_D = 4$ .

$$p = 21279538842228906832073178837320107985820544239452 \setminus$$

$$00643290551500599638430512859070665065630773920606 \setminus$$

$$859367176287315388271,$$

$$[\log_2 p] = 400,$$

$$a = 3,$$

$$b = 83049009004345361077229425543060980908577486512163 \setminus$$

$$20343513463682140711081496231001649308495612010228 \setminus$$

47712036885536216902,

$\#E(F_p) = 2127953884222890683207317883720107985820544239452 \setminus$

00643290553056368847309894942851909145900824563569,

612334595860183293074,

$= 22.9672517655558594014578717653327321811736611017 \setminus$

93273019677524116531294231770428569049611773102074 \setminus

34982378845266371967867.

EXAMPLE 11. CM method:  $D = 307$ , class number  $h_D = 3$ .

$p = 704884506943271274200281641864186967538228180387 \setminus$

43742878235725906364657764309029949371166271546975 \setminus

96008175843994317887,

$[\log_2 p] = 399.$

$a = 5,$

$b = 3866629042208848484615811897875529695758816114458122 \setminus$

72276326084773948335087614278974368305033461629194 \setminus 63497627079364752199,

$\#E(F_p) = 70488450694327127420028164186486186967538228180387 \setminus$

43742878233999375534968106454711645760031221836061

60284656185776243884  
= 4.17622112673581781855007041046621546741884557045 \  
09685935719558499843883742026613677911440007805459 \  
01540071164046444060971.

## Section 2. Characteristic Two

The examples in this section describe curves over fields  $F_q$  with  $q=2^n$ , defined by equations of the form

$$E: Y^2 + XY = X^3 + a_2 X^2 + a_6, \quad a_2, a_6 \in F_q.$$

For each curve, the values of  $n$ ,  $f(x)$ ,  $a_2$ ,  $a_6$ , and  $\#E(F_q)$  are listed, where  $f(x)$  is the irreducible polynomial used to represent  $F_q$  over  $F_2$ . The coefficient  $a_2$  is, in all the examples, either 0 or 1 and thus equal to its trace, as all values of  $n$  listed are odd. The coefficient  $a_6$  is presented in hexadecimal form. Each hexadecimal digit expands in the natural way to four bits, except possibly the most significant digit, which expands to the appropriate number of bits for a total length of  $n$ . Once expanded, the bits represent the coefficients of  $\alpha^{n-1}, \alpha^{n-2}, \dots, \alpha^0$ , respectively from left to right, where  $\alpha$  is a root of  $f(x)$ . The group order  $\#E(F_q)$  is shown in decimal form, and also factored as  $s \cdot r$ , where  $s$  is a small positive integer, and  $r$  is prime. In all the examples,  $s$  is the smallest possible value for the given isomorphism class, i.e.,  $s = 2$  when  $Tr_2(a_2) = 1$ ,  $s = 4$  otherwise. As before, a backslash at the end of a line indicates that the number (hexadecimal or decimal) is continued in the next line. All curves in this section are ‘random’.

EXAMPLE 12.  $n = 131$ ,  $f(x) = x^{131} + x^8 + x^3 + x^2 + 1$ ,

$$a_2 = 1,$$

$$a_6 = 7417501D24550DBC7735\ 1632C8513E8FE,$$

$$\# E(F_q) = 2722258935367507707729351292932711465734 \setminus$$

$$= 2.1361129467683753853864675646466355732867.$$

EXAMPLE 13.  $n = 131$ ,  $f(x) = x^{131} + x^8 + x^3 + x^2 + 1$ ,

$$a_2 = 0,$$

$$a_6 = 4AC7797773F8A77E6303D3D77655D6924,$$

$$\# E(F_q) = 272225893536750770780951877492775069508 \setminus$$

$$= 4.680564733841876926952379744373193767377.$$

EXAMPLE 14.  $n = 163$ ,  $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$ ,

$$a_2 = 1,$$

$$a_6 = 15E6478546D92CE2625DB7475B4368\ 9E6E40D4AD\ 4,$$

$$\# E(F_q) = 11692013098647223345629485326803604448910923041922 \setminus$$

$$= 2.58460065493236116728147426634018022244554615209 \setminus$$

61.

EXAMPLE 15.  $n = 163$ ,  $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$ ,

$$a_2 = 0,$$

$$a_6 = 48419ECBC9\ 470895FC140C851849CF\ 6F1977FF03B,$$

$$\begin{aligned} \#E(F_q) &= 11692013098647223345629482613505893115770279035908 \backslash \\ &= 4.29230032746618058364073706533764732789425697589 \backslash \end{aligned}$$

77.

EXAMPLE 16.  $n = 191$ ,  $f(x) = x^{191} + x^9 + 1$ ,

$$a_2 = 1,$$

$$a_6 = 7BC86E2102\ 902EC4D5890E8B6B4981\ FF27E0482750FE \backslash$$

$$FC03,$$

$$\#E(F_q) = 31385508676933403819178947116692299916305223017355 \backslash$$

$$90858398$$

$$= 2.15692754338466701909589473558346149958152611508 \backslash$$

$$67795429199.$$

EXAMPLE 17.  $n = 191$ ,  $f(x) = x^{191} + x^9 + 1$ ,

$$a_2 = 0,$$

$$a_6 = 315BB01ABA\ 43F4E87D289C59D9754AB5200A7 \setminus$$

$$489, \#E(F_q) = 31385508676933403819178947116561509135199454636589 \setminus$$

$$96854612$$

$$= 4.78463771692333509547947367791403772837998636591 \setminus$$

$$4749213653.$$

EXAMPLE 18.  $n = 239$ ,  $f(x) = x^{239} + x^{36} + 1$ ,

$$a_2 = 1,$$

$$a_6 = 6BAB7A91D4\ 794C8971A8\ 0A48B1DF53\ A464297EE0\ 89 \setminus$$

$$6C2EB097D93E4F0,$$

$$\#E(F_q) = 88342353238919216479164875037145925915902909620654 \setminus$$

7800094565304029091086

= 2.44171176619459608239582437518572962957951454810 \

3273900047282652014545543.

EXAMPLE 19.  $n = 239$ ,  $f(x) = x^{239} + x^{36} + 1$ ,

$a_2 = 0$ ,

$a_6 = 52BCEACD14\ FB3DCBCE42\ 1A3C6E59D4\ B663215EFF\ 1457 \setminus$

$498E4ABB6412CFA5$ ,

$\#E(F_q) = 88342353238919216479164875037145925936882090437622 \setminus$

7800094565304029091086

= 4.22085588309729804119791218759286481484220522609 \

4056232820608824169154047.

EXAMPLE 20.  $n = 307$ ,  $f(x) = x^{307} + x^8 + x^4 + x^2 + 1$ ,

$a_2 = 1$ ,

$a_6 = 393C7F7D53666B5054B5\ E6C6D3DE94\ F4296C0C599E2E \setminus$

$2E241050DF18B6090BDC90186904968BB$ ,



$\# E(F_q) = 26074060497081421904236104811640040461458795438640 \setminus$

$6558546126511192321845986215018738661814126$

$= 2.13037030248540710952118052405820020230729397719 \setminus$

$3203279273063255596160922993107509369330907063.$

EXAMPLE 21.  $n = 367$ ,  $f(x) = x^{367} + x^{21} + 1$ ,

$$a_2 = 1,$$

$a_6 = 43FC8AD242B0B7A6F3D1\ 627AD5654447556B47BF\ 6AA4 \setminus$

$A64B0C2AFE42CADAB8F93D92394C79A79755437B5699 \setminus$

$5136, \# E(F_q) = 30061345059505065316985351638903513950408736626026 \setminus$

$49434804528587635001816368941330023634165866357513 \setminus$

$18745406098,$

$= 2.15030672529752532658492675819451756975204368313 \setminus$

$01324717402264293817500908184470665011817082933178.$

$75659372703049.$

EXAMPLE 22.  $n = 401$ ,  $f(x) = x^{401} + x^{152} + 1$ ,

$$a_2 = 1,$$

$$a_6 = 83420635F8EA519BEC74\ 3DF9DBCA94\ AC950E076F\ 90C0\backslash$$

$$7C28212662E3C180FF8A2D2F4AF6DF2FB1833EFCEE99E\backslash$$

$$811CFB11CFA0,$$

$$\#E(F_q) = 51644997561738171793118383440060237486594115856584\backslash$$

$$47025661319699242150715677450218885459984002546145\backslash\ 032989725132571785934,$$

$$= 2.25822498780869085896559191720030118743297057928\backslash$$

$$29223512830659849621075357838725109442729992001273.$$

$$072516494862566285892967.$$

EXAMPLE 23.  $n = 431$ ,  $f(x) = x^{431} + x^{120} + 1$ ,

$$a_2 = 1,$$

$$a_6 = 715C87C2294703FF4B46C0BC257F89\ AE9E420BF6\ F07D\backslash$$

$$1E80A537F7269DAE06D7CD9EDECBCCF777D7D041F888\backslash$$

$$9D5C51A61C93DCC266CE,$$

$\# E(F_q) = 554533938824162971915682861674068728741507516 \backslash$   
 $3315034095916117808908283429806884365866090618516 \backslash$   
 $771699707619208876544223742366$   
 $= 2.277266969412081485957841430837034364370753 \backslash$   
 $75816575170479580585904454141714903442182933045309.$   
 $258385849853809604438272111871183.$

## Appendix B List of My Publications and Patents

---

### Publications

1. Pritam Gajkumar Shah, Xu Huang, Dharmendra Sharma, “Algorithm based on one’s complement for fast scalar multiplication in ECC for wireless sensor network,” The 3<sup>rd</sup> International Workshop on RFID & WSN and its Industrial Applications, in conjunction with IEEE AINA 2010, April 20-23, 2010, Perth, Australia. Paper ID-3.
2. Pritam Gajkumar Shah, Xu Huang, Dharmendra Sharma, “Analytical study of implementation issues of elliptical curve cryptography for wireless sensor networks,” The 3<sup>rd</sup> International Workshop on RFID & WSN and its Industrial Applications, in conjunction with IEEE AINA 2010, April 20-23, 2010, Perth, Australia. Paper ID-6.
3. Pritam Gajkumar Shah, Xu Huang, Dharmendra Sharma, “Sliding window method with flexible window size for scalar multiplication on wireless sensor network nodes,” International Conference on Wireless Communication and Sensor Computing (ICWCSC 2010), January 02-04, 2010, Chennai, Tamilnadu, India. Accepted and to be published.
4. Pritam Gajkumar Shah, “Network Security Protocols for Wireless Sensor Networks - A Survey”, International Conference on Cognitive Systems ICCS, 2005, New Delhi India.
5. Pritam Gajkumar Shah, “Performance Evaluation of Classics Flooding and Gossiping Algorithm for Wireless Sensor Networks”, IEEE cosponsored National Conference on Signal and Processing in Mumbai, India 2007.
6. Xu Huang, Pritam Gajkumar Shah, and Dharmendra Sharma, “Protecting from attacking the man-in-middle in wireless sensor networks with elliptic curve

cryptography key exchange,” 4<sup>th</sup> International Conference on Network and System Security, NSS 2010, Melbourne, Australia, September 1-3, 2010. Accepted and to be published.

7. Xu Huang, Pritam Gajkumar Shah and Dharmendra Sharma, “Multi-Agent System Protecting from Attacking with Elliptic Curve Cryptography,” the 2<sup>nd</sup> International Symposium on Intelligent Decision Technologies, Baltimore, USA, 28-30 July 2010. Accepted to be published.
8. Xu Huang, Pritam Shah, and Dharmendra Sharma, “Minimizing hamming weight based on 1’s complement of binary numbers over GF (2m),” IEEE 12<sup>th</sup> International Conference on Advanced Communication Technology, Phoenix Park, Korea Feb 7-10, 2010. ISBN 978-89-5519-146-2, pp.1226-1230.
9. Xu Huang, Pritam Shah, and Dharmendra Sharma, “Fast Algorithm in ECC for Wireless Sensor Network,” The International MultiConference of Engineers and Computer Scientists 2010, Hong Kong, 17-19 March 2010. Proceeding 818-822.
10. Xu Huang, Pritam Gajkumar Shah, and Dharmendra Sharma, “Fast scalar multiplication for elliptic curve cryptography in sensor networks with hidden generator point,” CyberC 2010: International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, October 10-12, Huangshan, China. Accepted and to be published.

## Patents

---

1. Australian Innovative Patent Number 2009101242 , “An apparatus and method for recoding of scalar based on one's complement subtraction for fast scalar multiplication in Elliptical Curve Cryptography for Wireless Sensor Network platform” , invented by Sharma, Dharmendra; Shah, P. G.; Huang, Xu, sealed on 18/12/2009.
2. Australian Innovative Patent Number 2010100259, “An apparatus and method for Sliding window method with dynamic window size for scalar multiplication on wireless sensor network nodes”, invented by Shah, Pritam G; Huang, Xu ; and Sharma, Dharmendra , sealed on 22/03/2010.
3. Australian Innovative Patent Number 2010100508, “Apparatus and method based on hidden generator point of elliptical curve cryptography for wireless sensor networks”, invented by Huang Xu; Shah, Pritam G; and Sharma Dharmendra, sealed on 26/05/2010.

## Appendix C Results Obtained On MIRACL Crypto Library

---

```
<240, 232> <242, 72> <242, 191> <245, 48> <245, 215> <246, 130>
<246, 133> <249, 98> <249, 165> <251, 75> <251, 188> <253, 116>
<253, 147> <254, 29> <254, 234> <259, 14> <259, 249> <260, 51>
<260, 212>

some point P = <1, 23>, 2P = <87, 61>
some point Q = <1, 240>, P+Q = <0, 0>
P += Q = <0, 0>
P += P = 2P = <87, 61>

EC message encryption example
=====
G = <148, 27>, order(G) is 15
Alice's public key Pa=a*G 51*<148, 27> = <3, 89>
Bob's public key Pb =b*G 212*<148, 27> = <61, 52>
Jane's public key Pj =j*G 153*<148, 27> = <195, 256>

Plain text message from Alice to Bob <m1,m2>: <19, 72>
Encrypted message from Alice to Bob = <Pa=a*G,c1=a*Pb*m1,c2=a*Pb*M2> = <<3, 89>,
193, 153>

      Bob's decrypted message from Alice<m1=C1/(b*Pa),m2=C2/(b*Pa)> = <19, 72>
Jane's decrypted message from Alice = <203, 191>
Press any key to continue . . .
```

An Elliptic Curve cryptography example  
by Jarl Ostensen modified by Pritam Shah ,Xu Huang,Dharmendra Sharma 2010

The elliptic curve over finite field and having equation:  
 $y^2 \bmod 263 = (x^3 + 1x + 1) \bmod 263$

Points on the curve (i.e. the group elements):

(0, 1) (0, 262) (1, 23) (1, 240) (2, 96) (2, 167)  
(3, 89) (3, 174) (4, 73) (4, 190) (6, 111) (6, 152)  
(7, 80) (7, 183) (8, 28) (8, 235) (10, 119) (10, 144)  
(14, 38) (14, 225) (15, 32) (15, 231) (21, 128) (21, 135)  
(22, 64) (22, 199) (23, 57) (23, 206) (24, 35) (24, 228)  
(27, 81) (27, 182) (29, 111) (29, 152) (30, 87) (30, 176)  
(31, 34) (31, 229) (33, 27) (33, 236) (38, 101) (38, 162)  
(39, 45) (39, 218) (40, 55) (40, 208) (44, 65) (44, 198)  
(45, 35) (45, 228) (47, 81) (47, 182) (48, 60) (48, 203)  
(49, 123) (49, 140) (51, 64) (51, 199) (57, 25) (57, 238)  
(58, 5) (58, 258) (59, 6) (59, 257) (60, 123) (60, 140)  
(61, 52) (61, 211) (66, 34) (66, 229) (67, 119) (67, 144)  
(68, 74) (68, 189) (69, 108) (69, 155) (72, 85) (72, 178)  
(74, 4) (74, 259) (75, 25) (75, 238) (77, 116) (77, 147)  
(78, 53) (78, 210) (81, 0) (82, 27) (82, 236) (83, 114)  
(83, 149) (87, 61) (87, 202) (88, 91) (88, 172) (99, 79)  
(99, 184) (103, 131) (103, 132) (108, 83) (108, 180) (110, 130)  
(110, 133) (111, 77) (111, 186) (112, 44) (112, 219) (115, 59)  
(115, 204) (116, 125) (116, 138) (117, 18) (117, 245) (118, 106)  
(118, 157) (123, 23) (123, 240) (127, 2) (127, 261) (131, 25)  
(131, 238) (132, 70) (132, 193) (134, 29) (134, 234) (137, 107)  
(137, 156) (138, 29) (138, 234) (139, 23) (139, 240) (141, 109)  
(141, 154) (142, 26) (142, 237) (143, 37) (143, 226) (144, 69)  
(144, 194) (145, 115) (145, 148) (146, 97) (146, 166) (147, 94)  
(147, 169) (148, 27) (148, 236) (150, 129) (150, 134) (152, 124)  
(152, 139) (154, 123) (154, 140) (157, 100) (157, 163) (159, 102)  
(159, 161) (162, 104) (162, 159) (166, 34) (166, 229) (169, 107)  
(169, 156) (170, 130) (170, 133) (173, 90) (173, 173) (174, 109)  
(174, 154) (175, 20) (175, 243) (180, 122) (180, 141) (181, 59)  
(181, 204) (182, 110) (182, 153) (184, 43) (184, 220) (185, 127)  
(185, 136) (186, 119) (186, 144) (188, 70) (188, 193) (189, 81)  
(189, 182) (190, 64) (190, 199) (192, 15) (192, 248) (194, 35)  
(194, 228) (195, 7) (195, 256) (196, 116) (196, 147) (199, 2)  
(199, 261) (200, 2) (200, 261) (206, 70) (206, 193) (207, 117)  
(207, 146) (208, 24) (208, 239) (210, 79) (210, 184) (211, 109)  
(211, 154) (216, 4) (216, 259) (217, 79) (217, 184) (218, 108)  
(218, 155) (219, 118) (219, 145) (220, 107) (220, 156) (221, 33)  
(221, 230) (222, 58) (222, 205) (223, 50) (223, 213) (224, 9)  
(224, 254) (226, 99) (226, 164) (228, 111) (228, 152) (230, 59)  
(230, 204) (236, 4) (236, 259) (239, 108) (239, 155) (240, 31)



```

571 bit ECDH :-
    offline, no precomputation      189.26 ms
    offline, w. precomputation      267.29 ms
    online                          189.26 ms
571 bit ECDSA :-
    signature no precomputation      189.26 ms
    signature w. precomputation      267.29 ms
    verification                    278.19 ms

571 bit GF(2^m) Koblitz Elliptic Curve....
ER -    164 iterations              61.26 ms per iteration
ED -    60 iterations              166.93 ms per iteration
EP -    36 iterations              281.25 ms per iteration

571 bit ECDH :-
    offline, no precomputation      61.26 ms
    offline, w. precomputation      281.25 ms
    online                          61.26 ms
571 bit ECDSA :-
    signature no precomputation      61.26 ms
    signature w. precomputation      281.25 ms
    verification                    166.93 ms
Press any key to continue . . .

```

```

283 bit ECDH :-
    offline, no precomputation      41.73 ms
    offline, w. precomputation      47.77 ms
    online                          41.73 ms
283 bit ECDSA :-
    signature no precomputation      41.73 ms
    signature w. precomputation      47.77 ms
    verification                     55.03 ms

283 bit GF(2^m) Koblitz Elliptic Curve....
ER -      888 iterations            11.26 ms per iteration
ED -      372 iterations            26.92 ms per iteration
EP -      206 iterations            48.69 ms per iteration

283 bit ECDH :-
    offline, no precomputation      11.26 ms
    offline, w. precomputation      48.69 ms
    online                          11.26 ms
283 bit ECDSA :-
    signature no precomputation      11.26 ms
    signature w. precomputation      48.69 ms
    verification                     26.92 ms

571 bit GF(2^m) Elliptic Curve....
ER -       40 iterations            253.50 ms per iteration
ED -       32 iterations            319.81 ms per iteration
EP -       36 iterations            281.25 ms per iteration

571 bit ECDH :-
    offline, no precomputation      253.50 ms
    offline, w. precomputation      281.25 ms
    online                          253.50 ms
571 bit ECDSA :-
    signature no precomputation      253.50 ms
    signature w. precomputation      281.25 ms
    verification                     319.81 ms

571 bit GF(2^m) Koblitz Elliptic Curve....
ER -      207 iterations            48.39 ms per iteration
ED -       70 iterations            143.74 ms per iteration
EP -       37 iterations            271.97 ms per iteration

571 bit ECDH :-
    offline, no precomputation      48.39 ms
    offline, w. precomputation      271.97 ms
    online                          48.39 ms
571 bit ECDSA :-
    signature no precomputation      48.39 ms
    signature w. precomputation      271.97 ms
    verification                     143.74 ms

```

224 bit ECDH :-

|                            |          |
|----------------------------|----------|
| offline, no precomputation | 38.02 ms |
| offline, w. precomputation | 15.90 ms |
| online                     | 38.02 ms |

224 bit ECDSA :-

|                             |          |
|-----------------------------|----------|
| signature no precomputation | 38.02 ms |
| signature w. precomputation | 15.90 ms |
| verification                | 48.31 ms |

256 bit GF(p) Elliptic Curve....

|      |                |                        |
|------|----------------|------------------------|
| ER - | 203 iterations | 49.33 ms per iteration |
| ED - | 165 iterations | 60.61 ms per iteration |
| EP - | 494 iterations | 20.28 ms per iteration |

256 bit ECDH :-

|                            |          |
|----------------------------|----------|
| offline, no precomputation | 49.33 ms |
| offline, w. precomputation | 20.28 ms |
| online                     | 49.33 ms |

256 bit ECDSA :-

|                             |          |
|-----------------------------|----------|
| signature no precomputation | 49.33 ms |
| signature w. precomputation | 20.28 ms |
| verification                | 60.61 ms |

163 bit GF(2<sup>m</sup>) Elliptic Curve....

|      |                |                        |
|------|----------------|------------------------|
| ER - | 813 iterations | 12.30 ms per iteration |
| ED - | 637 iterations | 15.70 ms per iteration |
| EP - | 598 iterations | 16.72 ms per iteration |

163 bit ECDH :-

|                            |          |
|----------------------------|----------|
| offline, no precomputation | 12.30 ms |
| offline, w. precomputation | 16.72 ms |
| online                     | 12.30 ms |

163 bit ECDSA :-

|                             |          |
|-----------------------------|----------|
| signature no precomputation | 12.30 ms |
| signature w. precomputation | 16.72 ms |
| verification                | 15.70 ms |

163 bit GF(2<sup>m</sup>) Koblitz Elliptic Curve....

|      |                 |                        |
|------|-----------------|------------------------|
| ER - | 2200 iterations | 4.55 ms per iteration  |
| ED - | 994 iterations  | 10.06 ms per iteration |
| EP - | 635 iterations  | 15.75 ms per iteration |

Elliptic Curve point multiplication benchmarks - calculating r.P  
 From these figures it should be possible to roughly estimate the time  
 required for your favourite EC PK algorithm, ECDSA, ECDH, etc.

Key - ER - Elliptic Curve point multiplication r.P  
       ED - Elliptic Curve double multiplication r.P + s.Q  
       EP - Elliptic Curve multiplication with precomputation  
 EC   - Elliptic curve GF(p) - p of no special form  
 ECDH - Diffie Hellman Key exchange  
 ECDSA - Digital Signature Algorithm

#### 160 bit GF(p) Elliptic Curve....

|      |                 |                        |
|------|-----------------|------------------------|
| ER - | 471 iterations  | 21.23 ms per iteration |
| ED - | 373 iterations  | 26.94 ms per iteration |
| EP - | 1014 iterations | 9.86 ms per iteration  |

#### 160 bit ECDH :-

|                            |          |
|----------------------------|----------|
| offline, no precomputation | 21.23 ms |
| offline, w. precomputation | 9.86 ms  |
| online                     | 21.23 ms |

#### 160 bit ECDSA :-

|                             |          |
|-----------------------------|----------|
| signature no precomputation | 21.23 ms |
| signature w. precomputation | 9.86 ms  |
| verification                | 26.94 ms |

#### 192 bit GF(p) Elliptic Curve....

|      |                |                        |
|------|----------------|------------------------|
| ER - | 337 iterations | 29.72 ms per iteration |
| ED - | 271 iterations | 37.01 ms per iteration |
| EP - | 775 iterations | 12.90 ms per iteration |

#### 192 bit ECDH :-

|                            |          |
|----------------------------|----------|
| offline, no precomputation | 29.72 ms |
| offline, w. precomputation | 12.90 ms |
| online                     | 29.72 ms |

#### 192 bit ECDSA :-

|                             |          |
|-----------------------------|----------|
| signature no precomputation | 29.72 ms |
| signature w. precomputation | 12.90 ms |
| verification                | 37.01 ms |

#### 224 bit GF(p) Elliptic Curve....

|      |                |                        |
|------|----------------|------------------------|
| ER - | 263 iterations | 38.02 ms per iteration |
| ED - | 207 iterations | 48.31 ms per iteration |
| EP - | 629 iterations | 15.90 ms per iteration |

|  |                             |                        |
|--|-----------------------------|------------------------|
| 224 bit ECDH :-                            |                             |                        |
|  | offline, no precomputation  | 12.56 ms               |
|  | offline, w. precomputation  | 15.70 ms               |
|  | online                      | 12.56 ms               |
| 224 bit ECDSA :-                           |                             |                        |
|  | signature no precomputation | 12.56 ms               |
|  | signature w. precomputation | 15.70 ms               |
|  | verification                | 16.42 ms               |
| 256 bit GF(p) Elliptic Curve....           |                             |                        |
| ER -                                       | 643 iterations              | 15.55 ms per iteration |
| ED -                                       | 480 iterations              | 20.83 ms per iteration |
| EP -                                       | 487 iterations              | 20.53 ms per iteration |
| 256 bit ECDH :-                            |                             |                        |
|  | offline, no precomputation  | 15.55 ms               |
|  | offline, w. precomputation  | 20.53 ms               |
|  | online                      | 15.55 ms               |
| 256 bit ECDSA :-                           |                             |                        |
|  | signature no precomputation | 15.55 ms               |
|  | signature w. precomputation | 20.53 ms               |
|  | verification                | 20.83 ms               |
| 163 bit GF(2^m) Elliptic Curve....         |                             |                        |
| ER -                                       | 755 iterations              | 13.25 ms per iteration |
| ED -                                       | 565 iterations              | 17.70 ms per iteration |
| EP -                                       | 593 iterations              | 16.86 ms per iteration |
| 163 bit ECDH :-                            |                             |                        |
|  | offline, no precomputation  | 13.25 ms               |
|  | offline, w. precomputation  | 16.86 ms               |
|  | online                      | 13.25 ms               |
| 163 bit ECDSA :-                           |                             |                        |
|  | signature no precomputation | 13.25 ms               |
|  | signature w. precomputation | 16.86 ms               |
|  | verification                | 17.70 ms               |
| 163 bit GF(2^m) Koblitz Elliptic Curve.... |                             |                        |
| ER -                                       | 1544 iterations             | 6.48 ms per iteration  |
| ED -                                       | 734 iterations              | 13.62 ms per iteration |
| EP -                                       | 571 iterations              | 17.51 ms per iteration |

|  |                             |                        |
|--|-----------------------------|------------------------|
| 163 bit ECDH :-  |                             |                        |
|  | offline, no precomputation  | 4.55 ms                |
|  | offline, w. precomputation  | 15.75 ms               |
|  | online                      | 4.55 ms                |
| 163 bit ECDSA :-                                       |                             |                        |
|  | signature no precomputation | 4.55 ms                |
|  | signature w. precomputation | 15.75 ms               |
|  | verification                | 10.06 ms               |
| 233 bit GF(2 <sup>m</sup> ) Elliptic Curve....         |                             |                        |
| ER -   | 365 iterations              | 27.40 ms per iteration |
| ED -   | 288 iterations              | 34.83 ms per iteration |
| EP -   | 326 iterations              | 30.72 ms per iteration |
| 233 bit ECDH :-  |                             |                        |
|  | offline, no precomputation  | 27.40 ms               |
|  | offline, w. precomputation  | 30.72 ms               |
|  | online                      | 27.40 ms               |
| 233 bit ECDSA :-                                       |                             |                        |
|  | signature no precomputation | 27.40 ms               |
|  | signature w. precomputation | 30.72 ms               |
|  | verification                | 34.83 ms               |
| 233 bit GF(2 <sup>m</sup> ) Koblitz Elliptic Curve.... |                             |                        |
| ER -   | 1284 iterations             | 7.79 ms per iteration  |
| ED -   | 526 iterations              | 19.01 ms per iteration |
| EP -   | 300 iterations              | 33.38 ms per iteration |
| 233 bit ECDH :-  |                             |                        |
|  | offline, no precomputation  | 7.79 ms                |
|  | offline, w. precomputation  | 33.38 ms               |
|  | online                      | 7.79 ms                |
| 233 bit ECDSA :-                                       |                             |                        |
|  | signature no precomputation | 7.79 ms                |
|  | signature w. precomputation | 33.38 ms               |
|  | verification                | 19.01 ms               |
| 283 bit GF(2 <sup>m</sup> ) Elliptic Curve....         |                             |                        |
| ER -   | 240 iterations              | 41.73 ms per iteration |
| ED -   | 182 iterations              | 55.03 ms per iteration |
| EP -   | 210 iterations              | 47.77 ms per iteration |
| 283 bit ECDH :-  |                             |                        |
|  | offline, no precomputation  | 41.73 ms               |
|  | offline, w. precomputation  | 47.77 ms               |
|  | online                      | 41.73 ms               |
| 283 bit ECDSA :-                                       |                             |                        |
|  | signature no precomputation | 41.73 ms               |

Elliptic Curve point multiplication benchmarks - calculating r.P  
 From these figures it should be possible to roughly estimate the time  
 required for your favourite EC PK algorithm, ECDSA, ECDH, etc.

Key - ER - Elliptic Curve point multiplication r.P  
       ED - Elliptic Curve double multiplication r.P + s.Q  
       EP - Elliptic Curve multiplication with precomputation  
 EC   - Elliptic curve GF(p) - p of no special form  
 ECDH - Diffie Hellman Key exchange  
 ECDSA - Digital Signature Algorithm

160 bit GF(p) Elliptic Curve....

|      |                 |                        |
|------|-----------------|------------------------|
| ER - | 1241 iterations | 8.06 ms per iteration  |
| ED - | 947 iterations  | 10.56 ms per iteration |
| EP - | 958 iterations  | 10.44 ms per iteration |

160 bit ECDH :-

|                            |          |
|----------------------------|----------|
| offline, no precomputation | 8.06 ms  |
| offline, w. precomputation | 10.44 ms |
| online                     | 8.06 ms  |

160 bit ECDSA :-

|                             |          |
|-----------------------------|----------|
| signature no precomputation | 8.06 ms  |
| signature w. precomputation | 10.44 ms |
| verification                | 10.56 ms |

192 bit GF(p) Elliptic Curve....

|      |                |                        |
|------|----------------|------------------------|
| ER - | 978 iterations | 10.22 ms per iteration |
| ED - | 732 iterations | 13.66 ms per iteration |
| EP - | 794 iterations | 12.59 ms per iteration |

192 bit ECDH :-

|                            |          |
|----------------------------|----------|
| offline, no precomputation | 10.22 ms |
| offline, w. precomputation | 12.59 ms |
| online                     | 10.22 ms |

192 bit ECDSA :-

|                             |          |
|-----------------------------|----------|
| signature no precomputation | 10.22 ms |
| signature w. precomputation | 12.59 ms |
| verification                | 13.66 ms |

224 bit GF(p) Elliptic Curve....

|      |                |                        |
|------|----------------|------------------------|
| ER - | 796 iterations | 12.56 ms per iteration |
| ED - | 609 iterations | 16.42 ms per iteration |
| EP - | 637 iterations | 15.70 ms per iteration |

256 bit GF(p) Elliptic Curve....

## **Appendix D Sample Source Codes Written in C, C++ for ECC**

---