CHAPTER 1: INTRODUCTION

A **Wireless Sensor Network** (WSN) consists of spatially distributed autonomous sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants. The developments of wireless sensor networks was motivated by military applications such as battlefield surveillance and are now used in many industrial and civilian application areas, including industrial process monitoring and control, machine health monitoring, environment and habitat monitoring, specific healthcare applications, home automation, and the traffic control.

A sensor network normally constitutes a wireless ad-hoc network, meaning that each sensor supports a multi-hop routing algorithm where nodes function as forwarders, relaying data packets to a base station. In addition to one or more sensors, each node in a sensor network is typically equipped with a radio transceiver or other wireless communications device, a small microcontroller, and an energy source, usually a battery. A sensor node might vary in size from that of a shoebox down to the size of a grain of dust, although functioning "motes" of genuine microscopic dimensions have yet to be created. Size and cost constraints on sensor nodes result in corresponding constraints on the resources such as energy, memory, computational speed and the bandwidth

The security in the domain of wireless sensor networks is currently provided mostly through the symmetric key cryptography. The related proposed protocols in the literature are based on the idea of keys before the deployment of the wireless sensor network. However, due to the limitation on memory resources of wireless sensor nodes, these protocols are not able to achieve perfect security and also face a key management problem in large scale wireless sensor networks. On the other hand asymmetric key cryptography offers flexibility to the sensor node and clean interface for the security component in the corresponding sensor network.

Elliptic curve cryptography is a complex computational model and before its implementation for wireless sensor network platform, several parameters have to be carefully analysed and then selected. The stated research in this thesis proposes a novel elliptic curve cryptography - timestamp based mutual authentication and key management scheme for a particular session between any two corresponding participating sensor networks in a wireless sensor network and correspondingly surveys the complexities of ECC and RSA based asymmetric key cryptography techniques for the proposed scheme and investigates the implementations issues of ECC and RSA based proposed scheme on Language C and Java 6 SE in particular for the domain of wireless sensor networks.

Nevertheless the same mutual authentication and key management scheme for a particular session in wireless sensor networks can be extended efficiently for a multi-session scenario in domain of wireless sensor networks or in the wired or wireless ad-hoc networks.

1.1 Motivation

The thesis aims at designing an efficient Mutual Authentication and Key Management Scheme for WSNs. The proposed scheme inherits the characteristic security features of ECC and simultaneously overcomes the drawback of loosely synchronized local clocks in the domain of distributed WSNs deployed in a hostile environment, that too with the introduction of a unique Timestamp Mechanism. Generally the security in WSNs is provided mostly through symmetric key cryptography. The already proposed protocols in the literature are based on the idea of keys before the deployment of WSN. However, due to the limitation on memory resources of wireless sensor nodes, these proposed protocols are not able to achieve perfect security and also face a key management problem in large scale WSNs. On the other hand asymmetric key cryptography offers flexibility to node and clean interface for the security component in the sensor network. These limitations and constraints laid down the foundation for the proposed work and became the key motivating factor for designing a secure and a scalable Mutual Authentication and Key Management Scheme in the domain of WSNs. Further the limitations and the constraints on memory resources and the compromised processing performance parameters inspired and encouraged us to use Elliptic Curve Cryptography in domain of Wireless Sensor Networks.

Consideration of these facts including memory constraints and limitations of substantial processing overhead along with compromised security basically laid down the foundation of this idea of working on a novel Elliptic Curve Cryptography-Timestamp based Mutual Authentication and Key Management Scheme for WSNs and efficiently proposing and implementing it in Language C and Java 6 SE programming platform using standard paradigm in characteristic phases for collecting practical data and efficiently concluding the result for the final comparison.

1.2 Related Work:

There are a few works available in the field of mutual authentication and particularly in key management in WSNs along with the use of Elliptic Curve Cryptography within the same and consideration of the factors such as security, scalability, conformance, adaptability, reliability and less memory utilization plays a vital role in deciding the major characteristic features of any mutual authentication and key management scheme in WSNs. In the stated works various characteristic features have been taken into account along with the consideration of the protocol or scheme complexities involved in the processing, memory limitations, bandwidth constraints of each and every module of a Wireless Sensor Networks and also have analyzed the different Elliptic Curve Cryptography variations along with the new proposals for key exchange or sharing in that particular domain. Some of which are stated and analysed below:

Basically a **Wireless Sensor Network** (WSN) consists of spatially distributed autonomous sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants as stated in the papers[1][2][3].

In the recent past, WSNs have found their way into a wide variety of applications and systems with vastly varying requirements and characteristics. As a consequence, it is becoming increasingly difficult to discuss typical requirements regarding hardware issues and software support. Although the important consequences of the design space of the WSNs are discussed in the paper [12].

Apart from this, the researchers had presented a survey of state-of-the-art of architecture and node deployment in WSNs and characteristics of the environment in which the sensor networks may deploy in the paper [8]. Although a modular approach, where the individual components of a sensor node can be easily exchanged has been suggested but different points in the design space would be required for different implementations of the corresponding interfaces.

Security topology and perspective in WSNs have been discussed in the paper [10] and it has been clearly stated that while implementing security in the mutual authentication and key management scenario many factors in the form of benchmarks need to be met where some of these benchmarks are specific to domain of wireless sensor networks while others are security benchmarks specific to traditional domain. In accordance to this, the characteristics of a WSN have also been described in the paper [9].

Now based on the network structure of clustering, a secure, efficient and authenticated group key agreement protocol for WSNs have been proposed in the paper [6] by using node-ID and bilinear pairings. Also in order to provide security, scalability, and flexibility to a WSN, an efficient Timestamp Counter Mechanism has been suggested by the researchers in paper [13].

As we know that the security approaches in WSNs due to limitation of energy and resource are different from traditional protocols in current networks, so an efficient dynamic authentication protocol in the domain of WSNs has been suggested by the researchers in paper [14].

Although ECC is complex and there are a number of practical issues to be resolved when integrating the technology into WSNs security system but one must consider the performance overheads in terms of the time, memory and bandwidth penalty for the use of authentication and encryption/decryption in WSNs applications. The paper [7] provides a brief view into the complex topic of ECC and aims to convey the fact that ECC has enormous potential for WSNs because of its smaller key size and its high strength of security provided that proper algorithms have been used for scalar multiplication process.

Scalar multiplication is the operation in ECC which takes 80 % of key calculation time on wireless sensor network motes. The research proposed in the paper [11] aims to deliver an efficient algorithm based on 1's complement subtraction to represent scalar in scalar multiplication which offer less Hamming weight and will remarkably improve the computational efficiency of scalar multiplication.

1.3 Problem Statement:

The main motive behind the proposed scheme in this thesis is to design and in turn successfully implement and propose an efficient and secure mutual authentication and key management scheme based on Elliptic Curve Cryptography for Wireless Sensor Networks using an appropriate simulation environment. The dissertation comprises of an analytical survey of the complexities involved in processing of ECC arithmetic and its variations and simultaneously investigates the issues with different implementations of ECC on wireless sensor network platforms. The project is concluded with a critique of inadequacies and how the current research attempts to address some of them with a summary of some early and current results from the implementation of proposed research. This thesis is to design, develop and propose:

"An Elliptic Curve Cryptography–Timestamp based Mutual Authentication and Key Management Scheme for Wireless Sensor Networks."

Reason for the selection of Elliptic Curve Arithmetic in Proposed Scheme for WSNs:

Although there have been many efficient trusted server scheme, self enforcing scheme, and key pre-distribution schemes for general key agreement in Wireless Sensor Networks but the problem and limitation in terms of memory constraints and higher processing overhead along with insecure node communication posses a great threat to the reliability and conformity of Wireless Sensor Nodes in distributed environment. With ECC and its modified authentication and key exchange scheme, the proposed scheme can very efficiently and robustly be implemented in available EEPROM with minimum utilization of available ROM, so as a consequence of it no additional hardware component in the form of neither extra ROM and RAM or any Arithmetic Coprocessor is required to fast and secure authentication or key management in respective sensor nodes. So basically these important issues needed to be considered before proposing and implementing a new mutual authentication and key management scheme for WSNs and hence the use of ECC in proposed scheme is justified.

Reason for the introduction of Timestamp Mechanism in Proposed Scheme for WSNs:

The time stamp mechanism in the proposed scheme is introduced in order to improve the security, scalability and flexibility of sensor nodes in hostile, inaccessible and mission critical environment. It offers flexibility to the designer, who can choose to trade off between different kinds of limited and constrained resources. By introducing time stamp mechanism, we tend to make our scheme independent of synchronous or asynchronous behaviour of the local clocks of the corresponding sensor nodes in a hostile environment. By the virtue of time stamping the counter value at the respective sensor node is automatically synchronized. The introduction of this mechanism in proposed scheme also solves the problem of loosely synchronized clocks. The mechanism also provides the option of dynamic counter allocation that too with variable counter size which means less memory utilization and improved performance. In this way the timestamp mechanism justifies its use in the proposed scheme.

1.4 Scope of Work:

The work done in this thesis is able to clearly demonstrate the importance of public key cryptography and its efficiency in mutual authentication and key management scheme in domain of WSNs. Though the project follows a systematic, hierarchal, organised and a structured approach in proposing, demonstrating and implementing the vital statistics of the key management scheme in WSNs but simultaneously it exploits the vital and beneficial characteristic features of an elliptic curve arithmetic in that scenario.

The proposed work is confined to a single session establishment, mutual authentication, verification, acknowledgement and secure data exchange between two different participating sensor nodes of the corresponding WSNs at a particular instance of time. The proposed work can be very efficiently extended for a multisession establishment, mutual authentication, verification, acknowledgement and secure data exchange between two or more than two sensor nodes at a particular instance in WSNs.

Nevertheless the same mutual authentication and key management scheme for a particular session in WSNs can be extended efficiently for a multi-session scenario in domain of WSNs or in the wired or wireless ad-hoc networks that too in distributed environment.

The proposed scheme is independent of the local and global clock synchronous or asynchronous behaviour and is efficient and accurate in both the scenario. The scheme has elliptic curve session parameters and time stamping mechanism which in itself is independent from synchronous or asynchronous behaviour of the clocks and makes it unique and this unique feature can be very efficiently implemented in embedded systems also.

The scope of the proposed scheme ranges from research domain to practical environment where the WSNs play a vital role such as industrial process monitoring and control, machine health monitoring, environment and habitat monitoring, healthcare applications, home automation, and traffic control monitoring or surveillance in battle field and its data security is of prime concern to us considering the processing limitations and memory constraints on the usage of these.

1.5 Organization of Thesis

The remainder of this thesis is organized as follows:

Chapter 2 presents the historical work carried out in this area and the state of art in mutual authentication and key management in wireless sensor networks along with the possibilities of the use of an ECC based asymmetric key cryptographic techniques in the domain of WSNs for efficient and secure session establishment and secure communication through the correspondingly established channel. All the works taken from the literature and described in this chapter are related to the problem of efficiently utilizing the limited and constrained memory resources along with limited processing overhead with the possible use of ECC based key management protocol in that particular domain of WSNs.

Chapter 3 introduces the vital basic concepts of Elliptic Curve Arithmetic. It then describes typical properties of an asymmetric cryptographic system in the domain of ECC with some mathematical background details. From there it takes on to the standard already proposed and widely accepted ECC based key exchange and mutual authentication scheme along with encryption and decryption scheme. Finally it elucidates the concepts and a technicality involved in different variations of point multiplications in domain of ECC and briefly summarizes the complexities involved in these.

Chapter 4 describes in detail the proposed scheme and the associated methodology employed in order to establish a secure communication channel for a particular session. First it describes the architectural layout of the proposed scheme, followed by the terminology or notation used in describing the corresponding scheme for mutual authentication and key management in the domain of WSNs. It then explains the proposed scheme in detail with related verifications of the corresponding phases. Finally it highlights the significance of the characteristic phases in the proposed scheme.

Chapter 5 presents the security perspective and analysis of the proposed scheme in a systematic and organized manner highlighting the security analysis of the corresponding attacks on sensor nodes in the domain of WSNs and security analysis of the corresponding nodes in different distributed domain.

Chapter 6 describes the experimental study and the corresponding implementation details of the proposed scheme in an organised way. Several comparative charts have also been introduced in order to clearly demonstrate the efficiency of the proposed scheme as compared to different variants of asymmetric key cryptographic techniques including ECC.

Chapter 7 covers the conclusion of the proposed work done in the thesis and gives the conclusion remarks about the characteristic results achieved from the implementation of the proposed scheme.

Chapter 8 discusses the possibilities and the scope of the proposed work in future and briefly highlights the corresponding enhancements which can be made to the current work for extending the proposal in future research.

Chapter 9 enlists the references used throughout the thesis and in proposed scheme.

Appendix A enlists the source code of the characteristic phases in the proposed scheme.

1.6 Conclusion:

This chapter has discussed the overview of the entire thesis which in turn helps to analyze the thesis and to some extent describes what the proposed work is all about. Further Chapters will describe the proposed work in somewhat more detail and will give a rigorous analysis of the proposed scheme along with the statistical facts of the respective implementation results or findings of this thesis along with the advantages and the limitations of the proposed scheme in the domain of WSNs.

CHAPTER 2: LITERATURE SURVEY

2.1 Objective:

Literature review constitutes an important section of my thesis. So following are the objectives behind the literature survey conducted by me:

- Place each work in the context of its contribution to the understanding of the subject under review.
- Describe the relationship of each work to the others under consideration.
- Identify new ways to interpret, and shed light on any gaps in, previous research.
- Resolve conflicts amongst seemingly contradictory previous studies.
- Identify areas of prior scholarship to prevent duplication of effort.
- Point the way forward for further research.
- Place my original work in the context of existing literature.

2.2 State of Art:

Security, scalability, conformance, adaptability, reliability and less memory utilization are major features of any key management algorithm or protocol in Wireless Sensor Networks. In this literature survey we have considered the complexities ,limitations ,constraints of each and every module of any Sensor Network and also analyzed the different Elliptic Curve

Cryptography variations along with the new proposals for key exchange or sharing in that domain .

Basically this literature survey has analysis and rigorous coverage of mainly two domain specific research surveys and facts and features of both viz.

- Key Management in Wireless Sensor Networks.
- Elliptic Curve Cryptography.

2.2.1 Wireless Sensor Networks:

A **Wireless Sensor Network** (WSN) consists of spatially distributed autonomous sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants [1][2][3]. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance and are now used in many industrial and civilian application areas, including industrial process monitoring and control, machine health monitoring [4], environment and habitat monitoring, healthcare applications, home automation, and traffic control [2][5].

In addition to one or more sensors, each node in a sensor network is typically equipped with a radio transceiver or other wireless communications device, a small microcontroller, and an energy source, usually a battery. A sensor node might vary in size from that of a shoebox down to the size of a grain of dust, although functioning "motes" of genuine microscopic dimensions have yet to be created. The cost of sensor nodes is similarly variable, ranging from hundreds of dollars to a few pennies, depending on the size of the sensor network and the complexity required of individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and bandwidth [2].

A sensor network normally constitutes a wireless ad-hoc network, meaning that each sensor supports a multi-hop routing algorithm where nodes function as forwarders, relaying data packets to a base station.

2.2.1.1 The Design Space of Wireless Sensor Networks:

"*The design space of wireless sensor networks***.**" Issues by **Romer, K.; Mattern, F** [12]. In the paper it was being described that in the recent past, wireless sensor networks have found their way into a wide variety of applications and systems with vastly varying requirements and characteristics. As a consequence, it is becoming increasingly difficult to discuss typical requirements regarding hardware issues and software support. This is particularly problematic in a multidisciplinary research area such as wireless sensor networks, where close collaboration between users, application domain experts, hardware designers, and software developers is needed to implement efficient systems. In this paper they have discussed the consequences of this fact with regard to the design space of wireless sensor networks by considering its various dimensions. They had also justified the view by demonstrating that specific existing applications occupy different points in the design space.

There are several important consequences of the design space as discussed above. Clearly, a single hardware platform will most likely not be sufficient to support the wide range of possible applications. In order to avoid the development of application-specific hardware, it would be desirable, however, to have available a (small) set of platforms with different capabilities that cover the design space. A modular approach, where the individual components of a sensor node can be easily exchanged, might help to partially overcome this difficulty. Principles and tools for selecting suitable hardware components for particular applications would also be desirable. As far as software is concerned, the situation becomes even more complex. As with hardware, one could try to cover the design space with a (larger) set of different protocols, algorithms, and basic services. However, a system developer would then still be faced with the complexity of the design space, since each application would potentially require the use of software with different interfaces and properties. In conventional distributed systems, middleware has been introduced to hide such complexity from the software developer by providing programming abstractions that are applicable for a large class of applications. This raises the question of whether appropriate abstractions and middleware concepts can be devised that are applicable for a large portion of the sensor network design space. This is not an easy task, since some of the design space dimensions (e.g., network connectivity) are very hard to hide from the system developer. Moreover, exposing certain application characteristics to the system and vice versa is a key approach for achieving energy and resource efficiency in sensor networks. Even if the provision of abstraction layers is conceptually possible, it would often introduce significant resource overheads which are problematic in highly resource-constrained sensor networks. At the workshop mentioned in the paper, some possible directions were discussed for providing general abstractions despite these difficulties. One approach is the definition of common service interfaces independent of their actual implementation. The interfaces would, however, contain methods for exposing application characteristics to the system and vice versa. Different points in the design space would then require different implementations of these interfaces.

2.2.1.2 Architecture and Node deployment in WSN:

"A Survey of Architecture and Node deployment in Wireless Sensor Network." Issues by **Pradnya Gajbhiye, Anjali Mahajan** [8]." In this paper it was being stated that Wireless Sensor Networks consist of small nodes with sensing, computation and wireless communication capabilities. Various architectures and node deployment strategies have been developed for wireless sensor network, depending upon the requirement of application, environmental monitoring, habitat monitoring, home automation, military application etc. In this paper they had presented a survey of state-of-the-art of architecture and node deployment in wireless sensor network. They also presented the characteristics of the environment in which the sensor networks may deploy. Node deployment in wireless sensor network is application dependent and can be either deterministic or randomized. They also explained the routing protocols for wireless sensor network.

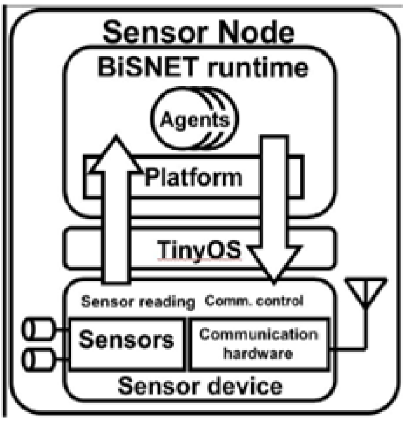


Figure 2.1: BiSNET Architecture

2.2.1.3 Security Topology in Wireless Sensor Networks:

"Security topology in wireless sensor networks with routing optimization." Issues by **Ismail, M.; Sanavullah, M.Y**.[10]. In the paper it was being conveyed that multiple sensor nodes deployed in a common neighbourhood to sense an event and subsequently transmit sensed information to a remote processing unit or base station, has been the recent focus of research. Tiny sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of sensor networks based on collaborative effort of a large number of nodes. These numerous sensors are used (similar to different sensory organs in human beings) for delivering crucial information in real-time from environments and processes, where data collection is impossible previously with wired sensors.

Typically, wireless sensor networks are composed of low power sensor nodes and integrate general-purpose computing with heterogeneous sensing and wireless communication. Their emergence has enabled observation of the physical world at an unprecedented level of granularity. One of the most important components of a sensor node is the power unit and may be supported in most applications by a power scavenging unit such as solar cells. Hence, there is a major limitation in a wireless sensor networks, such as, the sensor nodes must consume extremely low power. Also, wireless networks are subject to various kinds of attacks and wireless communication links can be eavesdropped on without noticeable effort and communication protocols on all layers are vulnerable to specific attacks.

In contrast to wire-line networks, known attacks like masquerading, man-in-the-middle, and replaying of messages can easily be carried out. Hence, a fundamental issue in the design of wireless sensor networks is the reliability i.e. how long can the wireless sensor networks survive and how well are the wireless sensor networks recovery after the malicious attacks. In this context, in the proposed work, the power, mobility, and task management planes that can monitor the power, movement, and task distribution among the sensor nodes are proposed. These planes help the sensor nodes coordinate the sensing task and also lower the overall power consumption. In addition, a secure topology discovery algorithm has been proposed and its performance is studied for different types of node distributions. The proposed work has the development of architecture for secure communication in mobile wireless networks. The approach divides the network into clusters and implements a decentralized certification authority. Decentralization has been achieved using threshold cryptography and a network secret that is distributed over a number of nodes. While this basic idea had been proposed earlier partially, its application on a clustered network is a novelty.

2.2.1.4 Security Perspective in Wireless Sensor Networks:

"Wireless Sensor Networks - A Security Perspective." Issues by Hasan **Tahir, Syed Asim Ali Shah** [9]. In this paper they aimed to convey that WSNs are complex network structures due to limitations in resources, size and hostile deployment environments. While implementing security in the mutual authentication and key management scenario many factors in the form of benchmarks need to be met where some of these benchmarks are specific to domain of wireless sensor networks while others are security benchmarks specific to traditional domain. Some of the security benchmarks in domain of wireless sensor networks are summarized as follows:

- **Energy Efficiency:** Implementation of security protocols should be energy efficient and should not drain the energy from the sensor node.

- **Efficient Public Key Cryptography:** Due to the limitations in memory and the lack of intensive computations ECC variant of public key cryptography is the best option for providing smaller key size and high security as compared to the traditional private key cryptography paradigm.

- **Tamper Resistance:** Usually the sensors in a wireless sensor are designed to be small cheap devices so as a result of which they are not able to resist tampering which means the strength of the sensors lie in their component design. A small number of faulty or suspicious sensors can cause the failure of the whole network in a hostile environment.

- **Multi Layers of Defence for Sensor Network Protocol Stack:** As an attacker can launch attack on multiple layers of the sensor network protocol stack so there should be a separate and peculiar security mechanism for each layer of the protocol stack.

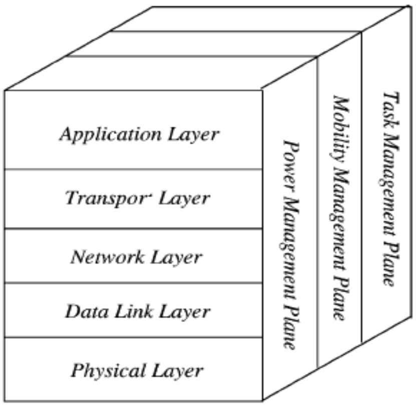
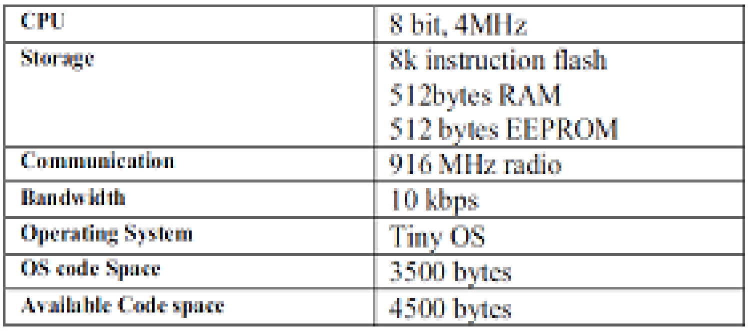


Figure 2.2: The Sensor Networks Protocol Stack

2.2.1.5 Characteristics of a Wireless Sensor Node:

"Wireless Sensor Networks - A Security Perspective." Issues by **Hasan Tahir, Syed Asim Ali Shah** [9]. In the paper it was being stated that Smart Dust Sensors are one of the most commonly used WSN in the fields of medical care, battlefield and environment monitoring, surveillance and disaster prevention and its characteristics are summarized in Table 2.1 which clearly depicts the resource constrained nature of sensor nodes. Many of these applications of the WSNS require that sensor node be deployed in an area that is hostile, inaccessible and mission critical. Keeping these important constraints and limitations into account the tasks like security management, resource consumption accounting, performance and failure management play a vital role in deciding & framing the overall architecture of an authentication or key management scheme. Lastly these characteristic features of a Sensor Node in a WSN, as stated in the paper are of prime concern when a proper layout of any mutual authentication or key management scheme in domain of WSNs is drafted as each and every resource constraint or limitation poses a serious challenge in front of the respective individual.



CPU	8 bit, 4MHz
Storage	8k instruction flash 512bytes RAM 512 bytes EEPROM
Communication	916 MHz radio
Bandwidth	10 kbps
Operating System	Tiny OS
OS code Space	3500 bytes
Available Code space	4500 bytes

Table 2.1: Characteristics of Smart Dust Sensors

2.2.1.6 A Novel Group Key Agreement Protocol for WSN:

"A Novel Group Key Agreement Protocol for Wireless Sensor Networks." Issues by **Zhang Li-Ping, Wang Yi and Li Gui-Ling** [6]. In the paper it was being described that recently, the wireless sensor networks has been used extensively in different domains. Form healthcare to warfare, and indoor spaces to outdoor ones, we can see the existence of sensor networks everywhere. But secure communication is a very important requirement in wireless sensor networks, and how to share the security group key between the sensor nodes becomes the major problem. In this paper, a secure, efficient and authenticated group key agreement protocol for wireless sensor networks is proposed by using node-ID and bilinear pairings. Comparing with previous group key management schemes for wireless sensor networks, their scheme is resistant against passive and active attacks, against node compromised attack, ensure the backward and forward security, and improve network computing complexity.

In this paper, based on the network structure of clustering, a secure, efficient and authenticated group key agreement protocol for wireless sensor networks is proposed by using node-ID and bilinear pairings. And it is being concluded after comparing with previous group key management schemes for wireless sensor network, their scheme can assure security against passive and active attacks, against node compromised attack, ensure the backward and forward security, and improve network computing complexity, but their scheme will add little communication burden, how to reduce the communication burden will be their further work.

2.2.1.7 Application of Time Stamp Mechanism on WSNs:

"Time-Stamp-Counter Mechanism and Application on Sensor Networks." Issues by **Zhan Liu and Mi Lu,** Texas A & M University College Station, USA [13]. In this paper, the researchers have proposed a time-stamp-counter mechanism which is applied to wireless sensor networks. This proposed mechanism has improved the security, scalability, and flexibility of sensor network. It also supports larger sensor networks. A designer can choose to trade off between many different kinds of resources. By applying this mechanism to SNEP as an example, it showed that their protocol could automatically synchronize counters, which makes the counter exchange protocol of SNEP an option not a must. Pair-wise counter is not a must. Instead, a dynamic counter allocation could be used, which reduces the number of counters. The mechanism also reduced the counter length. All of this resource reducing is important for resource limited networks, such as sensor networks and home networks.

2.2.1.8 Dynamic Authentication in Wireless Sensor Networks:

"Password Renewal Enhancement for Dynamic Authentication in Wireless Sensor Networks." Issues by **Farzad Kiani and Gokhan Dalkilic** Computer Engineering Department, Dokuz Eylul University, Izmir, Turkey [14]. In this paper they aimed to convey that security approaches in wireless sensor networks due to limitation of energy and resource are different from traditional protocols in current networks. So, many security problems exist such as authentication, integrity, digital signature and etc. and there has been relatively little research suited for wireless sensor networks. So they have proposed an enhanced version of scheme on dynamically clustering in wireless sensor network protocol. In comparison with the previous scheme, their proposed scheme possesses many advantages, including resistance to the replay and forgery attacks, reduction of user's password leakage risk, capability of renewing password and better efficiency.

2.2.2 Implementation Issues of Elliptic Curve Cryptography for WSNs:

"Analytical Study of Implementation Issues of Elliptic Curve Cryptography for Wireless Sensor Networks." Issues by **Pritam Gajkumar Shah, Xu Huang and Dharmendra Sharma** [7]. In the paper it was being described that Elliptic curve cryptography is a complex computational model and before its implementation for wireless sensor network platform, several parameters have to be carefully selected. This paper surveys the complexities of ECC and investigates issues with different implementations of ECC on wireless sensor network platforms. The paper concludes with a critique of inadequacies and how the current research attempts to address some of them with a summary of some early results from the research.

The security in wireless sensor networks is currently provided mostly through symmetric key cryptography. These protocols are based on the idea of keys before the deployment of the wireless sensor network. However, due to the limitation on memory resources of wireless sensor nodes, these protocols are not able to achieve perfect security and also face a key management problem in large scale wireless sensor networks. On the other hand asymmetric key cryptography offers flexibility to node and clean interface for the security component in the sensor network [10]. This research paper offered analytical study of implementation of ECC for wireless sensor networks.

2.2.3 Algorithm based on one's complement for fast Scalar Multiplication in ECC for Wireless Sensor Networks:

"Algorithm based on one's complement for fast scalar multiplication in ECC for Wireless Sensor Network"; Issues by **Pritam Gajkumar Shah, Xu Huang, Dharmendra Sharma** [11]. In this paper it was being described that Elliptic curve cryptography (ECC) is having good potential for wireless sensor network security due to its smaller key size and its high strength of security. But there is a room to reduce key calculation time to meet the potential applications in particular for wireless sensor networks. Scalar multiplication is the operation in elliptical curve cryptography which takes 80 % of key calculation time on wireless sensor

network motes. This research proposes algorithm based on 1's complement subtraction to represent scalar in scalar multiplication which offer less Hamming weight and will remarkably improve the computational efficiency of scalar multiplication.

The positive integer in point multiplication may be recoded with one' complement subtraction to reduce the computational cost involved in this heavy mathematical operation for wireless sensor network platforms. 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 inversion operation to get the NAF of binary number, the one's complement subtraction can provide a very simple way of recoding integer.

2.3 Conclusion of Literature Survey

From the whole literature survey I found that:

- There have been certain key management protocols in the past for WSNs but inefficient in assuring security and also incapable in providing efficient authentication and resulted in more processing overhead in terms of computational complexities and inefficient memory utilization.

- Elliptic Curve Cryptography can be a very good substitute for providing public key cryptography and can act as a remedy for providing a secured way of public key exchange in the domain of WSNs.

- Various enhancements can be made in scalar multiplication scheme in traditional ECDH, ECDSA and ECAES in order to enhance or speed up the computational overhead of Sensor Node.

- Lastly, there should be an efficient mechanism in the Key Management Scheme for WSNs for overcoming the drawback of loosely synchronized local clocks.

CHAPTER 3: BASICS OF ELLIPTIC CURVE CRYPTOGRAPHY

3.1 Mathematical Overview

3.1.1 Groups

A mathematical structure consisting of a set G and a binary operator $*$ on G is a group if,

- $\forall$a, b $\in$ G, if c = a $*$ b, then c $\in$ G (Closure)
- a $*$ (b $*$ c) = (a $*$ b) $*$ c, $\forall$a, b, c $\in$ G (Associative)
- $\exists$ e $\in$ G, such that $\forall$a $\in$ G, a $*$ e = e $*$ a = a (Identity element)
- $\forall$a $\in$ G, $\exists$ a′ $\in$ G such that, a $*$ a′ = a′ $*$ a = e. a′ is unique for each a and is called the inverse of a.

The group is represented as $\langle$G, $*\rangle$. Additionally, a group is said to be abelian if it also satisfies the commutative property, i.e., $\forall$a, b $\in$ G, if, a $*$ b = b $*$ a.

3.1.2 Rings

A Ring is a set R with two binary operations + and $*$ (Addition and multiplication) defined on R such that the following conditions are satisfied.

- $\langle$R, +$\rangle$ is an Abelian group
- a $*$ (b $*$ c) = (a $*$ b) $*$ c, $\forall$a, b, c $\in$ R (Associativity of $*$)
- a $*$ (b + c) = (a $*$ b) + (a $*$ c), $\forall$a, b, c $\in$ R (Distributivity of $*$ over +)

A Ring, in which $*$ is commutative is called a commutative ring. Further, if the ring contains an identity element with respect to $*$, i.e. $\exists$e $\in$ R and $\forall$ a $\in$ R, a $*$ e = e $*$ a = a, then e is called the identity element or the unity element and is represented by 1. If R contains a unity element, then R is called a Unitary Ring.

3.1.3 Fields and Vector Spaces

A Field F is a commutative and a unitary ring such that, F* = {a | a $\in$ F and a $\neq$ 0} is a multiplicative group. The ring Z_p is a Field, if and only if p is a prime.

If F is a field. A subset K of F that is also a field under the operations of F (with restriction to K) is called a sub field of F. In this case, F is called an extension field of K. If K $\neq$ F then K is a proper sub field of F. A field is called prime if it has no proper sub field.

If F is a field and V is an additive abelian group, then V is called the vector space over F, if an operation F x V → V is defined such that:

- *a (v + u) = av + au*
- *(a + b) v = av + bv*
- *a (bv) = (a.b) v*
- *1.v = v*

where, a, b $\in$ F and u, v $\in$ V.
The elements of F are called the scalars and the elements of V are called the vectors.

If v_1, v_2, …, v_m $\in$ V, and f_1, f_2, …, f_m $\in$ F, then the vector v' = $\sum f_i v_j$, $1 \le i, j \le m$, is a linear combination of the vectors in V. The set of all such linear combinations is called the **span** of V.

The vectors v_1, v_2, …, v_m $\in$ V are said to be linearly independent over F if there exists no scalars f_1, f_2, …, f_m $\in$ F such that $\sum f_i v_j \neq 0$, $1 \le i, j \le m$.

A set S = {u_1, u_2, …, u_n} are said to the basis of V iff all the elements of S are linearly independent and span V. If a vector space V over a field F has a basis of a finite number of vectors, then this number is called the dimension of V over F.

If F is an extension field of a field F_p then, F is a vector space over F_p. The dimension of F over F_p is called the degree of the extension of F over F_p.

3.1.4 Finite Fields

A field of a finite number of elements is denoted F_q or GF(q), where q is the number of elements. This is also known as a Galois Field.

The order of a Finite field F_q is the number of elements in F_q. Further, there exists a finite field F_q of order q iff q is a **prime power**, i.e. either q is prime or $q = p^m$, where p is prime. In the latter case, p is called the characteristic of F_q and m is called the extension degree of F_q and every element of F_q is a root of the polynomial $x^{p^m} - x$ over Z_p.

Let us consider two classes of Finite fields F_p (Prime Field, p is a prime number) and F_{2^m} (Binary finite field).

3.1.4.1 Prime Field F_p

The prime field F_p consists of the set of integers {0, 1, 2, ….., p – 1}, with the following arithmetic operations defined over it.

- **Addition:** $\forall a, b \in F_p, \exists r \in F_p$, where r = (a + b) mod p
- **Multiplication:** $\forall a, b \in F_p, \exists s \in F_p$, where s = (a $*$ b) mod p

3.1.4.2 Binary Finite Field F_{2m}

The finite field F_{2^m} , called a *characteristic two finite field* or a binary finite field can be viewed as a vector space of m dimensions over F_2, which consists of 2 elements 0 and 1. There exists m elements $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{m-1}$ in F_{2^m} such that each element $\alpha \in F_{2^m}$ can be uniquely represented as $\alpha = \sum_{i=0}^{m-1} a_i\alpha_i$, where $a_i \in \{0, 1\}, 0 \leq i < m$

The string $\{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{m-1}\}$ is called the basis of F_{2^m} over F_2. Given such a basis, every field element can be represented as a bit string ($a_0a_1a_2\ldots a_{m-1}$). Generally two kinds of basis are used to represent binary finite fields: polynomial basis and normal basis.

3.1.4.2.1 Polynomial basis representation of F_{2m}

Let $f(x) = x^m + f_{m-1}x^{m-1} + \dots + f_2x^2 + f_1x + f_0$, where $f_i \in \{0, 1\}$, $0 \leq i < m$, be an irreducible polynomial of degree m over F_2. $f(x)$ is called the reduction polynomial of F_{2^m}.

The finite field F_{2^m} is comprised of all polynomials over F2 of degree less than m, i.e.:

$F_{2^m} = \{a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \dots + a_2x^2 + a_1x + a_0 : a_i \in \{0, 1\}\}$.

The field element $a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \dots + a_2x^2 + a_1x + a_0$ is usually represented by the bit string $(a_{m-1}a_{m-2}\dots a_2a_1a_0)$ of length m such that

$F_{2^m} = \{(a_{m-1}a_{m-2}\dots a_2a_1a_0) : a_i \in \{0, 1\}\}$.

Thus, the elements of F_{2^m} can be represented by the set of all binary strings of length m. The multiplicative identity 1 is represented by the bit string (00…001) and the bit string of all zeroes represents the additive identity 0.

The following operations are defined on the elements of F_{2^m} when using $f(x)$ as the reduction polynomial.

- **Addition:** If a = $(a_{m-1}a_{m-2}\dots a_2a_1a_0)$ and b = $(b_{m-1}b_{m-2}\dots b_2b_1b_0)$ are elements of F_{2^m}, then, c = a + b = $(c_{m-1}c_{m-2}\dots c_2c_1c_0)$, where $c_i = (a_i + b_i) \bmod 2 = a_i \oplus b_i$.

- **Multiplication:** If a = $(a_{m-1}a_{m-2}\dots a_2a_1a_0)$ and b = $(b_{m-1}b_{m-2}\dots b_2b_1b_0)$ are elements of F_{2^m}, then, c = a . b = $(c_{m-1}c_{m-2}\dots c_2c_1c_0)$, where the polynomial $c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \dots + c_2x^2 + c_1x + c_0$ is the remainder when the polynomial $(a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \dots + a_1x + a_0)\,(b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \dots + b_1x + b_0)$ is divided by $f(x)$ over F_2.

- **Inversion:** If a is a nonzero element in F_{2^m}, then the inverse of a, denoted a^{-1}, is a unique element $c \in F_{2^m}$, where a.c = c.a = 1

3.1.4.2.2 Normal basis representation of F_{2m}

A normal basis of F_{2m} over F_2 is a basis of the form $\{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\}$, where $\beta \in F_{2m}$.

Any element $a \in F_{2m}$ can be written as $a = \sum_{i=0}^{m-1} a_i \beta^i$, where $a_i \in \{0, 1\}$.

3.1.4.2.3 Gaussian Normal Bases (GNB)

A GNB representation of F_{2m} exists if there exists a positive integer T such that p = Tm + 1 is prime and gcd(Tm/k , k) = 1, where k is the multiplicative order of 2 modulo p. The GNB representation is called a type T GNB for F_{2m}.

The following operations are defined over F_{2m} when using a type T GNB representation.

- **Addition:** If $a = (a_{m-1}a_{m-2}\ldots a_2a_1a_0)$ and $b = (b_{m-1}b_{m-2}\ldots b_2b_1b_0)$ are elements of F_{2m}, then, $c = a + b = (c_{m-1}c_{m-2}\ldots c_2c_1c_0)$, where $c_i = (a_i + b_i) \bmod 2 = a_i \oplus b_i$.
- **Squaring:** Let $a = (a_{m-1}a_{m-2}\ldots a_2a_1a_0) \in F_{2m}$. Squaring is a linear operation in F_{2m}. Hence $a^2 = \left(\sum_{i=0}^{m-1} a_i \beta^{2^i}\right)^2 = \sum_{i=0}^{m-1} a_i \beta^{2^{i+1}} = \sum_{i=0}^{m-1} a_{i-1} \beta^{2^i} = \left(a_{m-1}a_0a_2 \cdots a_{m-2}\right)$. Hence squaring a field element is simply a rotation of the vector representation.
- **Multiplication:** Let p = Tm + 1 and let $u \in F_p$. Let us define a sequence F(0), F(1), …, F(p - 1) by $F(2^i u^j \bmod p) = i$, for $0 \le i < m$, $0 \le j < T$.
  If $a = (a_{m-1}a_{m-2}\ldots a_2a_1a_0)$ and $b = (b_{m-1}b_{m-2}\ldots b_2b_1b_0)$ are elements of F_{2m}, then the product $c = a.b = (c_{m-1}c_{m-2}\ldots c_2c_1c_0)$ where,
  $$c_i = \begin{cases} \sum_{k=1}^{p-2} a_{F(k-1)+i}\, b_{F(p-k)+i} & \text{If T is even} \\ \sum_{k=1}^{m/2} \left(a_{k+i-1} b_{m/2+k+i-1} + a_{m/2+k+i-1} b_{k+i-1}\right) + \sum_{k=1}^{p-2} a_{F(k-1)+i}\, b_{F(p-k)+i} & \text{If T is odd} \end{cases}$$
  for each i, $0 \le i < m$, where indices are reduced modulo m.
- **Inversion:** If a is a nonzero element in F_{2m}, then the inverse of a, denoted a^{-1}, is a unique element $c \in F_{2m}$, where a.c = c.a = 1.

3.2 Elliptic Curves

Elliptic Curve Cryptography was introduced by Victor Miller and Neal Koblitz independently in the early eighties. The advantage of ECC over other public key cryptography techniques such as RSA is that the best known algorithm for solving ECDLP the underlying hard mathematical problem in ECC takes the fully exponential time and so far there is a lack of sub exponential attack on ECC.

3.2.1 Elliptic Curve Groups over Real Numbers

An elliptic curve over real numbers may be defined as the set of points (x,y) which satisfy an elliptic curve equation of the form:

$y^2 = x^3 + ax + b$, where x, y, a and b are real numbers.

Each choice of the numbers a and b yields a different elliptic curve. For example, a = -4 and b = 0.67 gives the elliptic curve with equation $y^2 = x^3 - 4x + 0.67$; the graph of this curve is shown below:

If $x^3 + ax + b$ contains no repeated factors, or equivalently if $4a^3 + 27b^2$ is not 0, then the elliptic curve $y^2 = x^3 + ax + b$ can be used to form a group. An elliptic curve group over real numbers consists of the points on the corresponding elliptic curve, together with a special point O called the point at infinity.

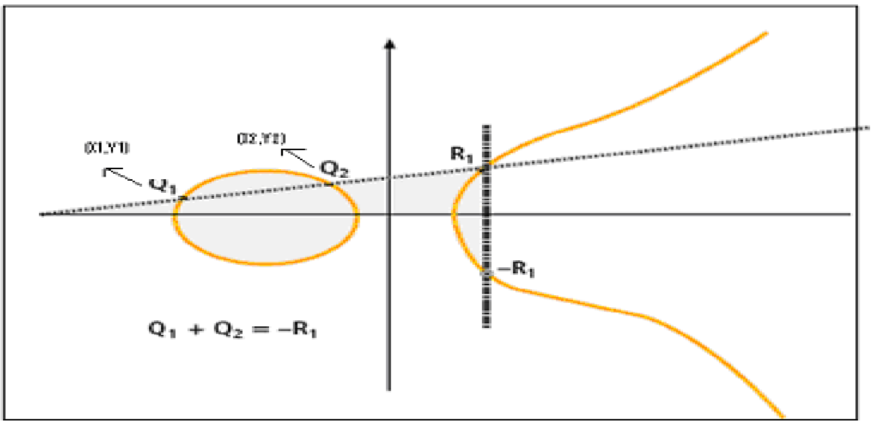


Figure 3.1: Elliptic Curve defined over Real Number.

3.2.1.1 Elliptic Curve Addition: A Geometric Approach

P + Q = R is the additive property defined geometrically.
Elliptic curve groups are additive groups; that is, their basic function is addition.
The addition of two points in an elliptic curve is defined geometrically.

The negative of a point P = (xP,yP) is its reflection in the x-axis: the point -P is (xP,-yP).
Notice that for each point P on an elliptic curve, the point -P is also on the curve.

3.2.1.1.1 Adding distinct points P and Q:

Suppose that P and Q are two distinct points on an elliptic curve, and the P is not -Q. To add the points P and Q, a line is drawn through the two points. This line will intersect the elliptic curve in exactly one more point, call -R. The point -R is reflected in the x-axis to the point R. The law for addition in an elliptic curve group is P + Q = R. For example:

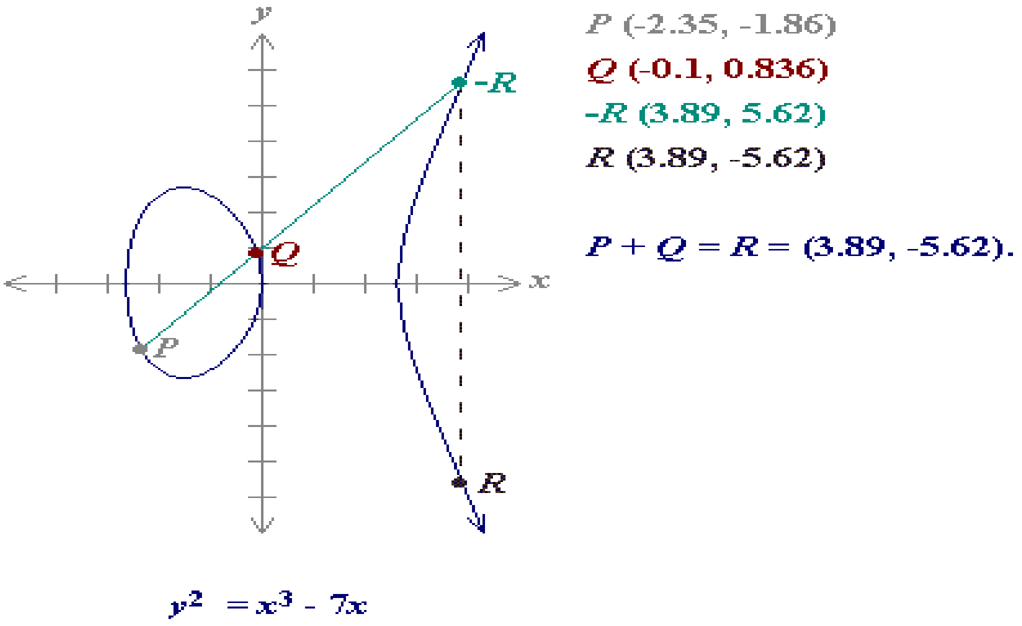


Figure 3.2: Addition of Points P & Q.

3.2.1.1.2 Adding the points P and –P:

The line through P and -P is a vertical line which does not intersect the elliptic curve at a third point; thus the points P and -P cannot be added as previously. It is for this reason that the elliptic curve group includes the point at infinity O. By definition, P + (-P) = O. As a result of this equation, P + O = P in the elliptic curve group . O is called the additive identity of the elliptic curve group; all elliptic curves have an additive identity.

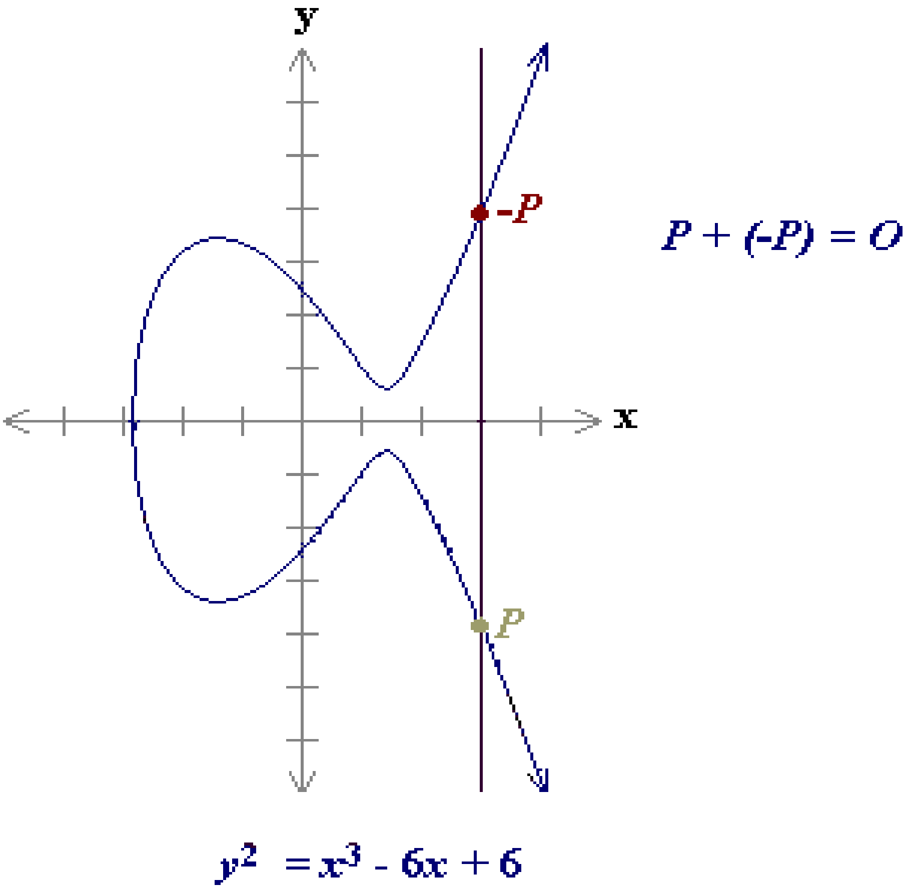


Figure 3.3: Addition of Points P & -P.

3.2.1.1.3 Doubling the point P:

To add a point P to itself, a tangent line to the curve is drawn at the point P. If yP is not 0, then the tangent line intersects the elliptic curve at exactly one other point, -R. -R is reflected in the x-axis to R. This operation is called doubling the point P; the law for doubling a point on an elliptic curve group is defined by:

P + P = 2P = R.

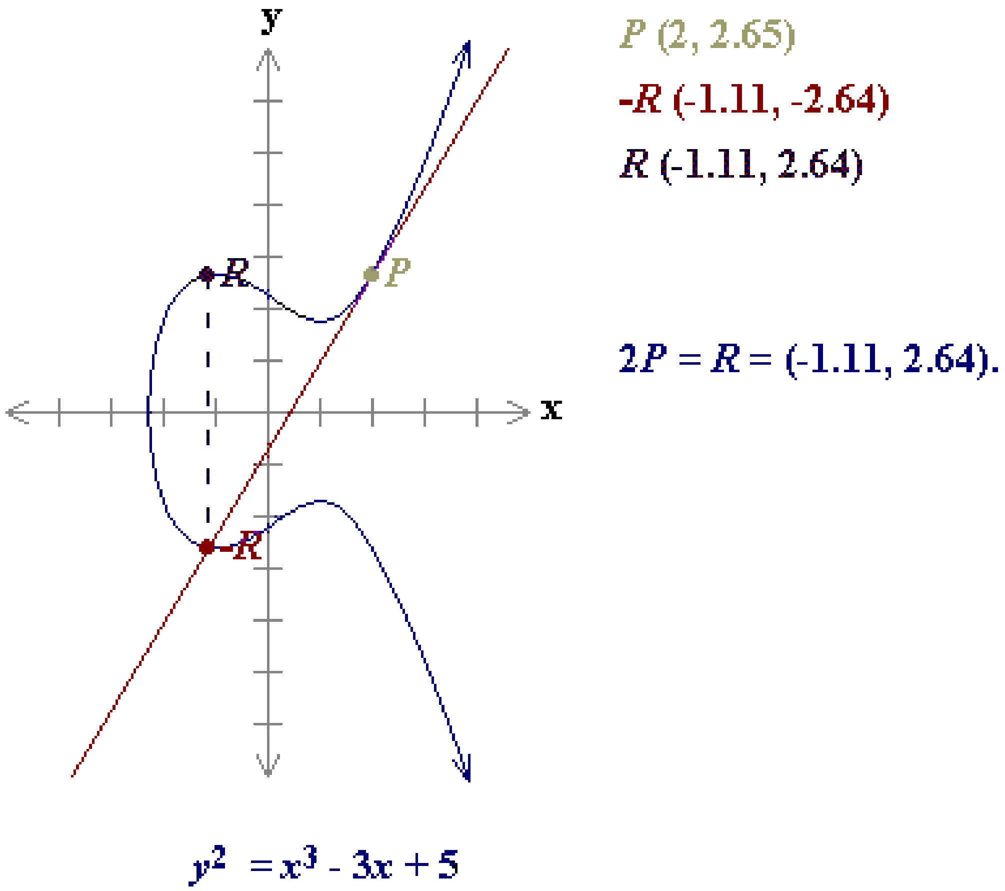


Figure 3.4: Doubling of Point P.

3.2.1.1.4 Doubling the point P if yP = 0:

The tangent from P is always vertical if

yP = 0.

If a point P is such that yP = 0, then the tangent line to the elliptic curve at P is vertical and does not intersect the elliptic curve at any other point.

By definition, 2P = O for such a point P.

If one wanted to find 3P in this situation, one can add 2P + P. This becomes P + O = P Thus 3P = P.

3P = P, 4P = O, 5P = P, 6P = O, 7P = P, etc.

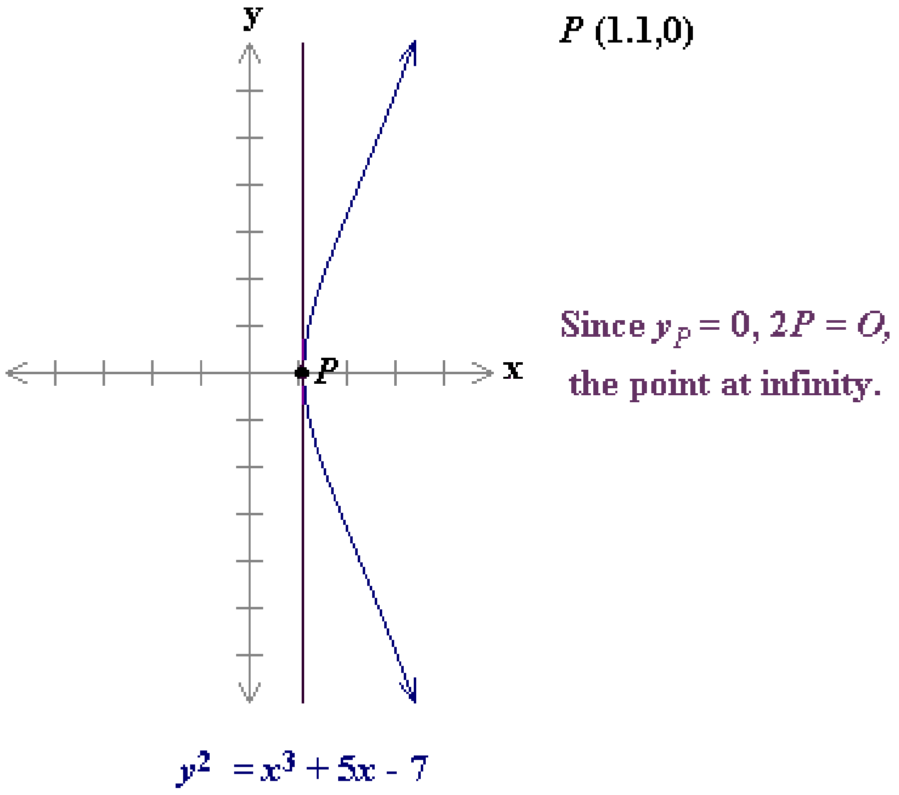


Figure 3.5: Doubling of Point P if yP=0.

3.2.1.2 Elliptic Curve Addition: An Algebraic Approach

Although the previous geometric descriptions of elliptic curves provide an excellent method of illustrating elliptic curve arithmetic, it is not a practical way to implement arithmetic computations. Algebraic formulae are constructed to efficiently compute the geometric arithmetic.

3.2.1.2.1 Adding distinct points P and Q:

When P = (x_P,y_P) and Q = (x_Q,y_Q) are not negative of each other,

P + Q = R where

$s = (y_P - y_Q) / (x_P - x_Q)$

$x_R = s^2 - x_P - x_Q$ and $y_R = -y_P + s(x_P - x_R)$

Note that s is the slope of the line through P and Q.

3.2.1.2.2 Doubling the point P:

When y_P is not 0,

2P = R where

$s = (3x_P^2 + a) / (2y_P)$

$x_R = s^2 - 2x_P$ and $y_R = -y_P + s(x_P - x_R)$

Recall that a is one of the parameters chosen with the elliptic curve and that s is the tangent on the point P.

3.2.2 Elliptic Curve Groups over F_p:

An Essential property for cryptography is that a group has a finite number of points. Calculations over the real numbers are slow and inaccurate due to round-off error. Cryptographic applications require fast and precise arithmetic; thus elliptic curve groups over the finite fields of F_p and F_2m are used in practice.

Recall that the field Fp uses the numbers from 0 to p - 1, and computations end by taking the remainder on division by p. For example, in F_{23} the field is composed of integers from 0 to 22, and any operation within this field will result in an integer also between 0 and 22.

An elliptic curve with the underlying field of Fp can formed by choosing the variables a and b within the field of F_p. The elliptic curve includes all points (x,y) which satisfy the elliptic curve equation modulo p (where x and y are numbers in F_p).

For example: y^2 mod p = x^3 + ax + b mod p has an underlying field of Fp if a and b are in Fp.

If x^3 + ax + b contains no repeating factors (or, equivalently, if $4a^3 + 27b^2$ mod p is not 0), then the elliptic curve can be used to form a group. An elliptic curve group over Fp consists of the points on the corresponding elliptic curve, together with a special point O called the point at infinity. There are finitely many points on such an elliptic curve.

There are several major differences between elliptic curve groups over F_p and over real numbers. Elliptic curve groups over F_p have a finite number of points, which is a desirable property for cryptographic purposes. Since these curves consist of a few discrete points, it is not clear how to "connect the dots" to make their graph look like a curve. It is not clear how geometric relationships can be applied. As a result, the geometry used in elliptic curve groups over real numbers cannot be used for elliptic curve groups over F_p. However, the algebraic rules for the arithmetic can be adapted for elliptic curves over F_p. Unlike elliptic curves over real numbers, computations over the field of F_p involve no round off error - an essential property required for a cryptosystem.

3.2.2.1 Adding distinct points P and Q:

The negative of the point P = (x_P, y_P) is the point -P = (x_P, -y_P mod p). If P and Q are distinct points such that P is not -Q, then

P + Q = R where
s = (y_P - y_Q) / (x_P - x_Q) mod p
$x_R = s^2 - x_P - x_Q$ mod p and $y_R = -y_P + s(x_P - x_R)$ mod p

Note that s is the slope of the line through P and Q.

3.2.2.2 Doubling the point P:

Provided that y_P is not 0,
2P = R where
$s = (3x_P^2 + a) / (2y_P) \bmod p$
$x_R = s^2 - 2x_P \bmod p$ and $y_R = -y_P + s(x_P - x_R) \bmod p$

Recall that a is one of the parameters chosen with the elliptic curve and that s is the slope of the line through P and Q.

3.2.3 Elliptic Curve Groups over F_2^m:

Elements of the field F_2^m are m-bit strings. The rules for arithmetic in F_2^m can be defined by either polynomial representation or by optimal normal basis representation. Since F_2^m operates on bit strings, computers can perform arithmetic in this field very efficiently.

An elliptic curve with the underlying field F_2^m is formed by choosing the elements a and b within F_2^m (the only condition is that b is not 0). As a result of the field F_2^m having a characteristic 2, the elliptic curve equation is slightly adjusted for binary representation:

$$y^2 + xy = x^3 + ax^2 + b$$

The elliptic curve includes all points (x,y) which satisfy the elliptic curve equation over F_2^m (where x and y are elements of F_2^m). An elliptic curve group over F_2^m consists of the points on the corresponding elliptic curve, together with a point at infinity, O. There are finitely many points on such an elliptic curve.

3.2.3.1 Arithmetic in an Elliptic Curve Group over F_2^m:

Elliptic curve groups over F_2m have a finite number of points, and their arithmetic involves no round off error. This combined with the binary nature of the field, F_2^m arithmetic can be performed very efficiently by a computer. The following algebraic rules are applied for arithmetic over F_2^m.

3.2.3.1.1 Adding distinct points P and Q:

The negative of the point P = (xP, yP) is the point -P = (xP, xP + yP). If P and Q are distinct points such that P is not -Q, then

P + Q = R where

s = (yP - yQ) / (xP + xQ)

xR = s^2 + s + xP + xQ + a and yR = s(xP + xR) + xR + yP

As with elliptic curve groups over real numbers, P + (-P) = O, the point at infinity.

Furthermore, P + O = P for all points P in the elliptic curve group.

3.2.3.1.2 Doubling the point P:

If xP = 0, then 2P = O

Provided that xP is not 0,

2P = R where

s = xP + yP / xP

xR = s^2+ s + a and yR = xP^2 + (s + 1) * xR

Recall that a is one of the parameters chosen with the elliptic curve and that s is the slope of the line through P and Q.

3.3 Elliptic Curve Groups and the Discrete Logarithm Problem:

At the foundation of every cryptosystem is a hard mathematical problem that is computationally infeasible to solve. The discrete logarithm problem is the basis for the security of many cryptosystems including the Elliptic Curve Cryptosystem. More specifically, the ECC relies upon the difficulty of the Elliptic Curve Discrete Logarithm Problem(ECDLP).

Recall that we examined two geometrically defined operations over certain elliptic curve groups. These two operations were point addition and point doubling. By selecting a point in a elliptic curve group, one can double it to obtain the point 2P. After that, one can add the point P to the point 2P to obtain the point 3P. The determination of a point nP in this manner is referred to as Scalar Multiplication of a point. The ECDLP is based upon the intractability of scalar multiplication products.

While it is customary to use additive notation to describe an elliptic curve group (as has been done previously in this classroom), some insight is provided by using multiplicative notation. Specifically, consider the operation called "scalar multiplication" under additive notation: that is, computing kP by adding together k copies of the point P. Using multiplicative notation, this operation consists of multiplying together k copies of the point P, yielding the point P*P*P*P&.*P = Pk.

3.3.1 The Elliptic Curve Discrete Logarithm Problem:

In the multiplicative group Zp*, the discrete logarithm problem is: given elements r and q of the group, and a prime p, find a number k such that r = qk mod p. If the elliptic curve groups is described using multiplicative notation, then the elliptic curve discrete logarithm problem is: given points P and Q in the group, find a number that Pk = Q; k is called the discrete logarithm of Q to the base P. When the elliptic curve group is described using additive notation, the elliptic curve discrete logarithm problem is: given points P and Q in the group, find a number k such that Pk = Q.

Example:

In the elliptic curve group defined by

$y^2 = x^3 + 9x + 17$ over F_{23},

What is the discrete logarithm k of Q = (4,5) to the base P = (16,5)?

One (naïve) way to find k is to compute multiples of P until Q is found. The first few multiples of P are:

P = (16,5) 2P = (20,20) 3P = (14,14) 4P = (19,20) 5P = (13,10) 6P = (7,3) 7P = (8,7) 8P = (12,17) 9P = (4,5)

Since 9P = (4,5) = Q, the discrete logarithm of Q to the base P is k = 9.

In a real application, k would be large enough such that it would be infeasible to determine k in this manner.

3.4 Application of Elliptic Curves in Key Exchange

3.4.1 Elliptic Curve Cryptography (ECC) domain parameters:

The public key cryptographic systems involves arithmetic operations on Elliptic curve over finite fields which is determined by elliptic curve domain parameters.

The ECC domain parameters over F_q is defined by the septuple as given below
D = (*q, FR, a, b, G, n, h*), where

- ***q***: prime power, that is q = p or q = 2^m, where p is a prime

- ***FR***: field representation of the method used for representing field elements $\in F_q$

- ***a, b***: field elements, they specify the equation of the elliptic curve E over F_q,
  $y^2 = x^3 + ax + b$
- ***G***: A base point represented by G= (x_g, y_g) on E (F_q)

- ***n***: Order of point G , that is n is the smallest positive integer such that nG = ***O***

- ***h***: cofactor, and is equal to the ratio $\#E(F_q)/n$, where $\#E(F_q)$ is the curve order

The primary security in ECC is the parameter n; therefore the length of ECC key is the bit length of n. For comparative length, the security of ECC keys is much more than that of other cryptosystems. That is for equivalent security, the key length of ECC key is much lesser than other cryptosystems.

3.4.2 Elliptic Curve protocols:

Generally in the process of encryption and decryption, we have 2 entities, the one at the encryption side and the other at the decryption side. Let us assume that Alice is the person who is encrypting and Bob is the person decrypting.

Key generation: Alice's (or Bob's) public and private keys are associated with a particular set of elliptic key domain parameters (q, FR, a, b, G, n, h).

Alice generates the public and private keys as follows

1. Select a random number d, $d \in [1, n - 1]$
2. Compare Q = dG.
3. Alice's public key is Q and private key is d.

It should be noted that the public key generated needs to be validated to ensure that it satisfies the arithmetic requirement of elliptic curve public key. A public key $Q = (x_q, y_q)$ associated with the domain parameters (q, FR, a, b, G, n, h) is validated using the following procedure:

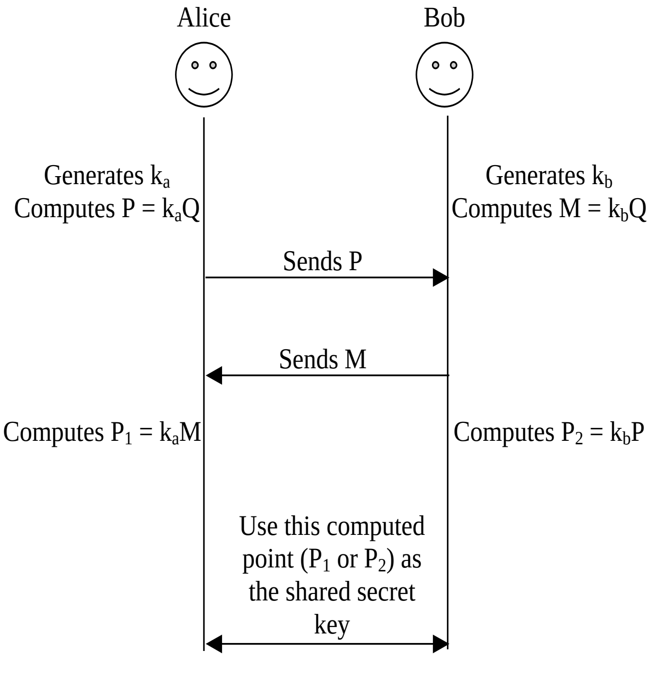
1. Check that $Q \neq \boldsymbol{O}$
2. Check that x_q and y_q are properly represented elements of F_q
3. Check if Q lies on the elliptic curve defined by a and b.
4. Check that $nQ = \boldsymbol{O}$

3.4.2.1 Elliptic Curve Diffie-Helman protocol (ECDH):

ECDH is elliptic curve version of Diffie-Hellman key agreement protocol. The protocol for generation of the shared secret using ECC is as described below:

- Alice takes a point Q and generates a random number k_a
- Alice computes the point P = k_a Q and sends it to Bob (It should be noted that Q, P are public)
- Bob generates a random number k_b and computes point M = k_b.Q and sends it to Alice
- Alice now computes $P_1 = k_aM$ and Bob computes $P_2 = k_bP$
- P1 = P2 = k_b k_bQ, this is used as the shared secret key

An illustration of the above steps is represented below.



Algorithm 3.1: Elliptic Curve Diffie-Hellman Protocol

3.4.2.2 Elliptic Curve Digital Signature Authentication (ECDSA):

Alice, with domain parameters D = (q, FR, a, b, G, n, h), public key Q and private key d, does the following steps to sign the message m

Step 1: Selects a Random number $k \in [1, n - 1]$

Step 2: Computes Point kG = (x, y) and r = x mod n, if r = 0 then goto Step 1

Step 3: Compute $t = k^{-1} \bmod n$

Step 4: Compute e = SHA-1(m), where SHA-1 denotes the 160 bit hash function

Step 5: Compute $s = k^{-1} (e + d_a{*}r) \bmod n$, if s = 0 goto Step 1

Step 6: The signature of message m is the pair (r, s)

Step 7: Alice sends Bob the message m and her signature (r, s)

To verify Alice's signature, Bob does the following (Note that Bob knows the domain parameters D and Alice's public key Q)

Step 1: Verify r and s are integers in the range [1, n – 1]

Step 2: Compute e = SHA-1(m)

Step 3: Compute $w = s^{-1} \bmod n$

Step 4: Compute $u_1 = e.w$ and $u_2 = r.w$

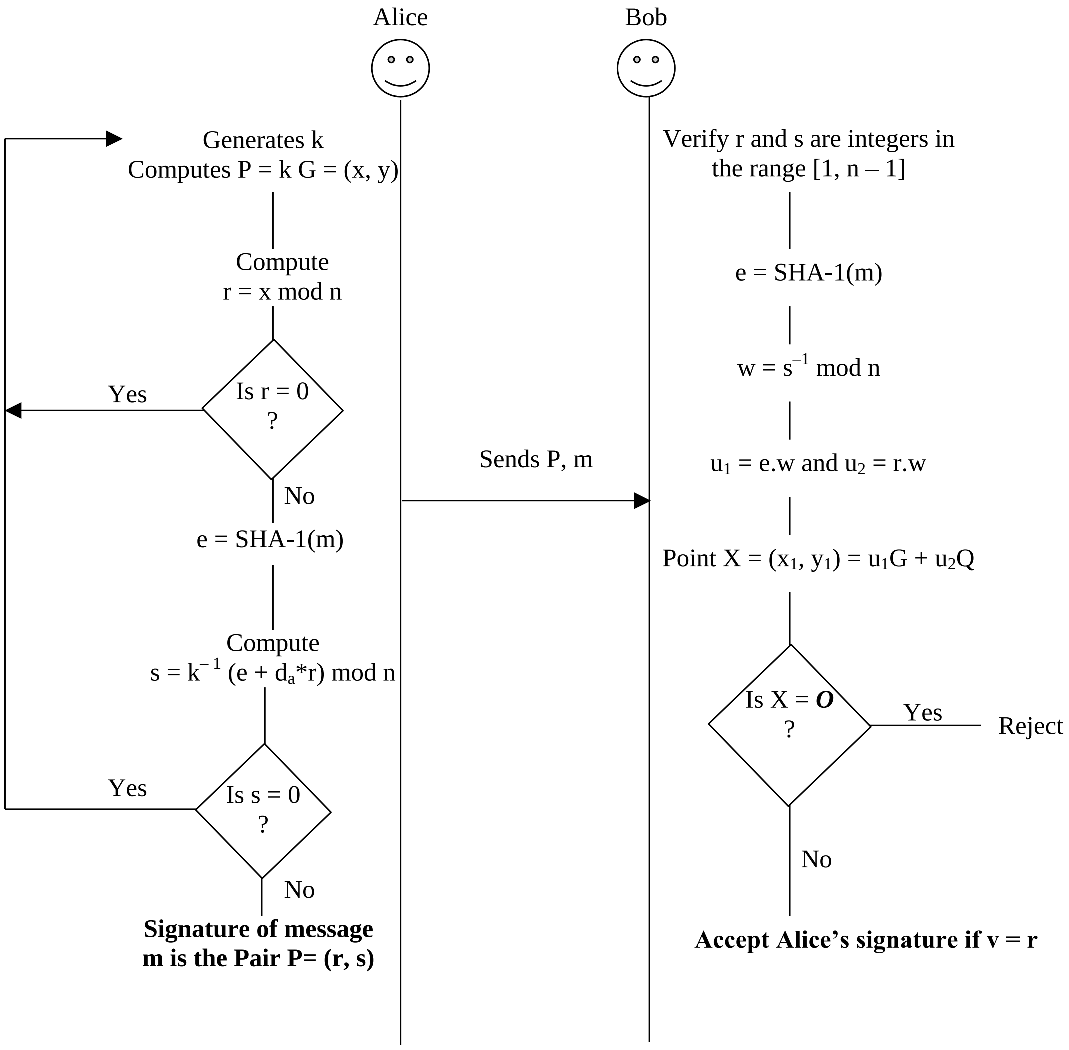
Step 5: Compute Point $X = (x_1, y_1) = u_1G + u_2Q$

Step 6: If $X = \boldsymbol{O}$, then reject the signature

Else compute $v = x_1 \bmod n$

Step 7: Accept Alice's signature iff v = r.

An illustration of the above steps is represented below:



Algorithm 3.2: Elliptic Curve Digital Signature Algorithm

Proof for verification:

If the message is indeed signed by Alice, then $s = k^{-1} (e + d*r) \bmod n$.

That is, $k = s^{-1} (e + d.r) \bmod n = s^{-1} e + s^{-1} d.r = w.e + w.d.r = (u_1 + u_2.d) \bmod n$ ……[1]

Now consider $u_1G + u_2Q = u_1G + u_2dG = (u_1 + u_2.d) G = kG$ from [1]

In step 5 of the verification process, we have $v = x_1 \bmod n$, where,

Point X = (x1, y1) = $u_1G + u_2Q$. Thus we see that v = r since r = x mod n and x is the x coordinate of the point kG and we have already seen that $u_1G + u_2Q = kG$.

3.4.2.3 Elliptic Curve Authentication Encryption Scheme (ECAES):

Alice has the domain parameters D = (q, FR, a, b, G, n, h) and public key Q. Bob has the domain parameters D. Bob's public key is Q_B and private key is d_B. The ECAES mechanism is as follows.

Alice performs the following stepsA does the following

Step 1: Selects a random integer r in [1, n – 1]

Step 2: Computes R = rG

Step 3: Computes $K = hrQ_B = (K_x, K_y)$, checks that $K \neq \boldsymbol{O}$

Step 4: Computes keys $k_1||k_2 = KDF(K_x)$ where KDF is a key derivation function, which derives cryptographic keys from a shared secret

Step 5: Computes $c = ENC_{k1}(m)$ where m is the message to be sent and ENC a symmetric encryption algorithm

Step 6: Compute $t = MAC_{k2}(c)$ where MAC is message authentication code

Step 7: Sends (R, c, t) to Bob

To decrypt a cipher text, Bob performs the following steps

Step 1: Perform a partial key validation on R (check if $R \neq \boldsymbol{O}$, check if the coordinates of R are properly represented elements in F_q and check if R lies on the elliptic curve defined by a and b)

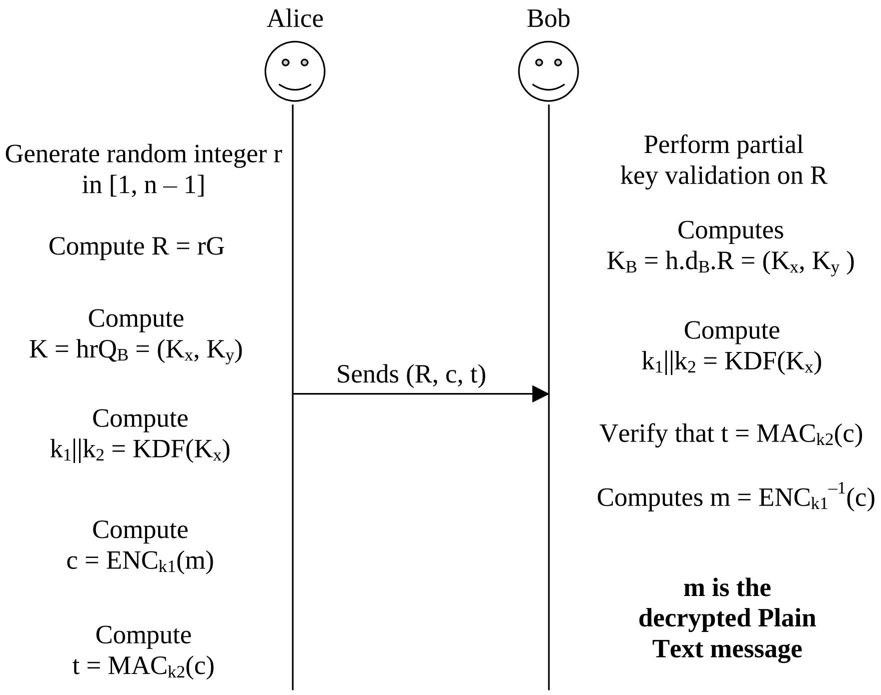
Step 2: Computes $K_B = h.d_B.R = (K_x, K_y)$, check $K \neq \boldsymbol{O}$

Step 3: Compute $k_1, k_2 = KDF (K_x)$

Step 4: Verify that $t = MAC_{k2}(c)$

Step 5: Computes $m = ENC_{K_1^{-1}}(c)$

We can see that $K = K_B$, since $K = h.r.Q_B = h.r.d_B.G = h.d_B.r.G = h.d_B.R = K_B$


Alice
Bob
Generate random integer r in [1, n – 1]
Compute R = rG
Compute $K = hrQ_B = (K_x, K_y)$
Compute $k_1||k_2 = KDF(K_x)$
Compute $c = ENC_{k1}(m)$
Compute $t = MAC_{k2}(c)$
Sends (R, c, t)
Perform partial key validation on R
Computes $K_B = h.d_B.R = (K_x, K_y)$
Compute $k_1||k_2 = KDF(K_x)$
Verify that $t = MAC_{k2}(c)$
Computes $m = ENC_{k1}^{-1}(c)$
m is the decrypted Plain Text message

Algorithm 3.3: Elliptic Curve Authentication Encryption Scheme

3.4.3 Algorithms for Elliptic Scalar Multiplication:

In all the protocols that were discussed (ECDH, ECDSA, ECAES), the most time consuming part of the computations are scalar multiplications. That is the calculations of the form

$$Q= k\,P = P + P + P\ldots \text{ k times}$$

Here P is a curve point, k is an integer in the range of order of P (i.e. n). P is a fixed point that generates a large, prime subgroup of $E(F_q)$, or P is an arbitrary point in such a subgroup. Elliptic curves have some properties that allow optimization of scalar multiplications. The following sections describe some efficient algorithms for computing kP.

3.4.3.1 Non adjacent form (NAF):

This is a much efficient method used in the computation of kP. Here, the integer k is represented as $k = \sum_{j=0}^{l-1} k_j 2^j$, where each kj $\in$ {–1, 0, 1}. The weight of NAF representation of a number of length *l* is *l*/3. Given below is an algorithm for finding NAF of a number.

```
NAF(k)
Comment: Returns u[] which contains the NAF representation of k
Begin
        c ← k
        l ← 0

        While (c > 0)
        BeginWhile
                If (c is odd)
                BeginIf
                        u[l] ← 2 – (c mod 4)
                        c ← c – u[l]
                Else
                        u[l] ← 0
                EndIf

                c ← c/2
                l ← l + 1
        EndWhile
        Return u
End
```

Algorithm 3.4: Computation of the NAF of a scalar

The generation of NAF for k = 7 = $(111)_2$ is as shown below:

No of iterations	c	*l*	u
1	7	0	-1
2	4	1	0
3	2	2	0
4	1	3	1

Table 3.1: Illustration of computation of NAF(7)

Therefore, the value of 7 in NAF form is (1 0 0 –1). (Note that no two consecutive digits are non-zero)

3.4.4 Complexity analysis of the Elliptic Scalar Multiplication Algorithms

3.4.4.1 Binary Method:

The simplest formula for calculating kP is based on the binary representation of k, i.e.,

$k = \sum_{j=0}^{l-1} k_j 2^j$, where $k_j \in \{1,0\}$, the value kP can be computed by

$$kP = \sum_{j=0}^{l-1} k_j 2^j . P = 2(...2(2k_{l-1.} P + k_{l-2} P) + ...) + k_{0.} P$$

This method requires *l* doublings and w_k-1 additions, where w_k (the weight) is the number of 1s in the binary representation of k.

For k = 7 = $(111)_2$, the value of kP would be

$$kP = \sum_{j=0}^{l-1} k_j 2^j P = 2(2.P + P) + 1P$$

3.4.4.2 Addition-Subtraction method:

Here the number k is represented in NAF form. The algorithm performs addition or subtraction depending on the sign of each digit, scanned from left to right.

The algorithm is as given below:

```
Addition-Subtraction ( k, P)
Comment: Return Q = kP, where Point P = (x, y) ∈ E(Fq)
Begin
        u[] ← NAF(k)  /* The NAF form of k is stored in u */
        Q ← O

        For j = l – 1 DownTo 0
        BeginFor
                Q ← 2Q

                If ($u_j$ = 1)
                Then

                ElseIf ($u_j$ = –1)
                Then
                        Q ← Q – P
                EndIf
        EndFor

        Return Q
End
```

Algorithm 3.5: Scalar Multiplication using the Addition-Subtraction method.

The algorithm performs *l* doublings and *l*/3 additions on an average.

For k = 7, the binary method would require 3 doublings and 3 additions.

In case of Addition-Subtraction method (the value of 7 in NAF form is 1 0 0 –1), it would require 4 doublings and 2 additions.

3.4.4.3 Repeated doubling method:

A point on the elliptic curve over F_{2m} is represented inn the form of (x, λ) rather than in the form of (x, y) when using the repeated doubling method for scalar multiplication. Every point $P = (x, y) \in E(F_{2m})$, where $x \neq 0$, P can be represented as the pair (x, λ), where $\lambda = x + y/x$. The algorithm is as given below:

```
Repeated-doubling(P, i)
Comment: Returns $Q = 2^iP$
Begin
    $\lambda \leftarrow x + y/x$
    For j = 1 to i – 1
    BeginFor
        $x_2 \leftarrow \lambda^2 + \lambda + a$
        $\lambda_2 \leftarrow \lambda^2 + a + b/( x^4 +   b)$
    EndFor
    $x_2 \leftarrow \lambda_2 + \lambda + a$
    $y_2 \leftarrow x_2 + (\lambda + 1)x_2$
    $Q \leftarrow (x_2, y_2)$
    Return Q
End
```

Algorithm 3.6: Scalar Multiplication using Repeated Additions.

CHAPTER 4: PROPOSED SCHEME

4.1 Introduction:

The proposed mutual authentication and key management scheme described below has been structured and organized in a systematic and hierarchal way and is composed of the different characteristic phases which in turn depends on the security of the elliptic curve primitives e.g. random key generation, time stamp and mutual signature generation or time stamp and mutual signature verification. The corresponding arithmetic operations on the elliptic curve points exploits the security factor associated with the elliptic curve equation defined over a finite field and the difficulty associated with finding the set of solutions in the domain of ECC arithmetic. The fundamental and the most important security factor associated with this proposed scheme depends on the intractability of the ECC analogue of Discrete Logarithm Problem, which in itself is a well known and extensively studied computationally hard problem. Simultaneously the use of Time Stamping mechanism improves the security, scalability and flexibility of WSNs.

4.2 Proposed Scheme:

Consider a scenario where a large number of autonomous wireless sensor nodes are spatially distributed and deployed in a hostile mission critical environment for some specific purpose of monitoring and in turn sharing critical information with the base station or the sink node and also with the participating sensor node in an established session for communication between the two respective participants.

Suppose we have a scenario where sensor nodes are organized in respective clusters as shown in Figure 4.1 and one node in each cluster acts as a cluster manager. The cluster manager is responsible for collecting information from the sensor nodes in its cluster and forwarding it to a sink node. The corresponding cluster manager is also responsible for securely distributing the unique IDs to its cluster nodes and also maintaining the database of those IDs.

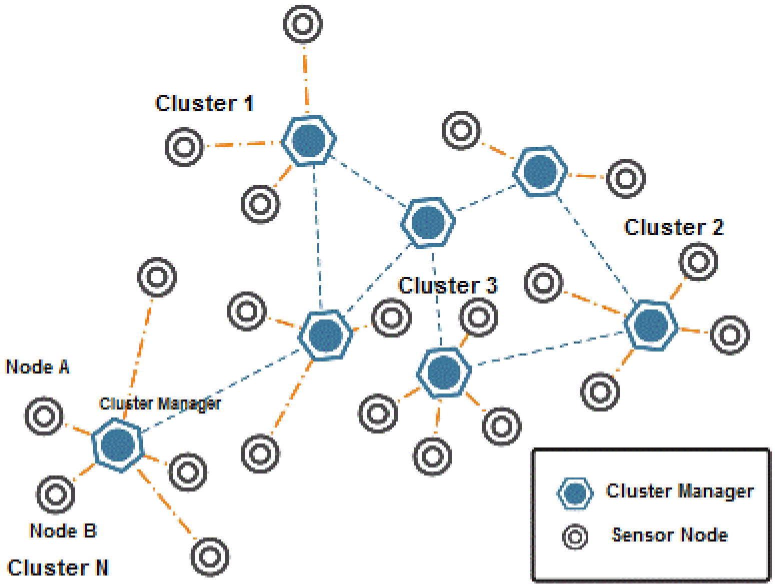


Figure 4.1 Clustered Architecture of a Wireless Sensor Network

Consider a Nth Cluster in a deployed Wireless Sensor Network where we two sensor nodes viz. Node A and Node B want to communicate and exchange some critical information among them. Before the establishment of session between the two, the Cluster Manager of that cluster securely distributes the unique IDs to the respective sensor nodes. After the distribution of IDs, the participating Sensor Node A generates the ECC domain parameters **D = (q, F_2^m, a, b, P, n, h, T_S)** over F_2^m and then make these parameters public by transmitting these parameters to another Sensor Node B in a secure manner. Once the ECC domain parameters are made public, the session among the respective participating nodes A & B is created and then the following characteristic phases are followed in a sequential manner and are described as follows:

4.2.1 Notation Used:

In this thesis, we use the following notation of the ECC domain parameters to describe our proposed scheme. The ECC domain parameters over $F_2{}^m$ is defined by the septuple as given below:

D = (q, $F_2{}^m$, a, b, P, n, h, T_S) , where

- **q** : prime power, that is $q = p$ or $q = 2^m$, where p is a prime.
- **$F_2{}^m$** : field representation of the method used for representing field elements $\in F_q$.
- **a, b** : field elements, they specify the equation of the elliptic curve E over F_q, $y^2 = x^3 + ax + b$.
- **P** : random base point represented by **$P = (x_p, y_p)$** on **$E (F_2{}^m)$.**
- **n** : Order of point P, where n is the smallest positive integer such that **$nP = O$** also satisfying the condition that n is a large prime number.
- **h** : cofactor, and is respectively equal to the ratio $\#E(F_2{}^m)/n$, where $\#E(F_2{}^m)$ is the curve order.
- **T_S**: the threshold session time interval for a particular session between two participants' viz. node A & B.

In the proposed work we use the following notation to describe our mutual authentication and key management scheme which includes notations for time stamping parameters and parameters used for elliptical curve cryptographic operations:

- A, B are participants, such as communicating nodes in a particular session.

- K_{ab} is the secret mutual agreed key, shared between the participating nodes A and B in a particular session for the purpose of sharing unique identities and the time stamps during the initial key exchange process where $K_{ab} = (K_x,K_y)$ is a base point representation on the elliptic curve E $(F_2{}^m)$.

- K_m is the secret mutual agreed key, shared between the participating nodes A and B in a particular session for the purpose of authenticating each other mutually where $K_m = (K'_x,K_y')$ is a base point representation on the elliptic curve E $(F_2{}^m)$.

- K_S is the common session key generated for the purpose of transmitting and receiving the encrypted data packets and the message blocks during the authentication encryption phase of the proposed scheme where $K_S = (K''_x,K_y'')$ is a base point representation on the elliptic curve E $(F_2{}^m)$.

- Q_K is defined as the common and intermediate mutually agreed parameter where $Q_K = d_a \times d_b \times P$.

- k_a , k_b and d_a , d_b are the pair of random numbers generated by participating nodes A and B respectively at their ends and d_a , d_b and k_a , $k_b \in [1, n - 1]$.

- KDF ( ) is the Key Derivation Function which derives cryptographic keys from a shared key.

- B is a bit variable which denotes “0” or “1” packet that means even or odd packet.

- T_i is the current time stamp of the corresponding participants in a session and is decided by the expression $T_i = T_0 + i.\Delta T$.

- T_0 is the starting time of a time interval of a particular session established between two participating nodes.

- Δt is the maximum local clock error in a participating node during session establishment.

- i, is a variable whose domain consists of integers
  i.e. i ϵ {-ϖ…..-2, -1 ,0, 1, 2,……+ϖ}

- ∆T is a time interval of a particular session established between two participating nodes, for the purpose of communication.

- T_{delay} is a time delay in arrival of packet from one participating node to another during communication in a particular session.

- C_{ab} is a counter maintained by a participating node A when a communication session is established between node A and another participating node B.

- C_{ba} is a counter maintained by a participating node B when a communication session is established between node B and another participating node A.

- ID_A and ID_B are the corresponding unique identities generated by the corresponding participating nodes A and B for secure communication in a session.

- x, denotes the secret unique password generated simultaneously by participating nodes A and B for secure communication in a session.

- ENC(k, M) and DEC(k, M) correspondingly denote the private key encryption and decryption of the message block M with the private key k.

- MAC_K (Message M) denotes the computation of Message Authentication Code of the corresponding Message Block M with key K.

- $T_i \parallel C_a$ means concatenation of the current time stamp of participating nodes and counter maintained by node A itself.

- SHA_1 (Message M) denotes a 160 bit hash function which computes the 160 bit hash value of the Message Block M.

4.2.2 Diagrammatic Representation of Proposed Scheme:

Phase I: Mutual Agreed Key Generation & Exchange

Node A		Node B
Session Initiates :		
Generates ECC domain parameters		
$D = (q, F_{2^m}, a, b, P, n, h, T_S)$ over F_{2^m}		
	$\xrightarrow{\text{Transmits}}$	
Chooses $k_a \in [1, n-1]$ Computes $Q_a = k_a.P$		Chooses $k_b \in [1, n-1]$ Computes $Q_b = k_b.P$
Sends	$\xrightarrow{Q_a}$	Receives
Receives	$\xleftarrow{Q_b}$	Sends
Computes $P_1 = k_a.\ Q_b = (X_a, Y_a)$		Computes $P_2 = k_b.\ Q_a = (X_b, Y_b)$
Computes $K_{ab} = P_1 = P_2 = k_a.k_b.P$ where $K_{ab} = (K_x,K_y)$		Computes $K_{ab} = k_a.k_b.P$ where $K_{ab} = (K_x,K_y)$
Checks if $P_1=P_2\neq O$		Checks if $P_1=P_2\neq O$
Computes $K_1 = KDF(K_x)$ and $K_2 = KDF(K_y)$		Computes $K_1 = KDF(K_x)$ & $K_2 = KDF(K_y)$

Phase II: Time Stamp Generation

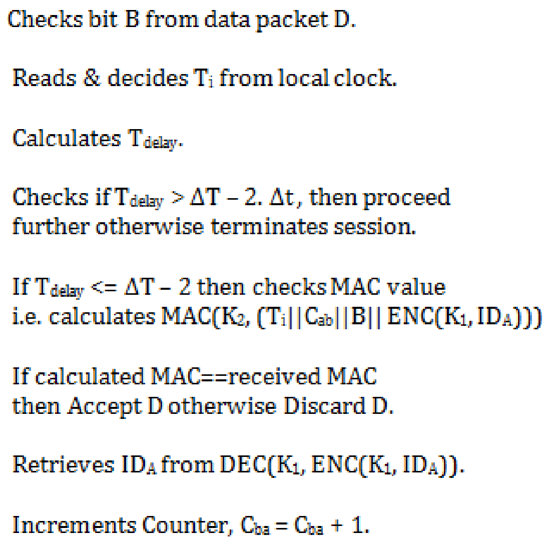
Node A	Node B
Chooses bit B to be “0” or “1”.	
Chooses T_i where $T_i = T_0 + i*\Delta T$.	

Phase III: Identity Signature Generation & Exchange

Node A	Node B
Generates D where	
$D = (B \,\|\|\, ENC(K_1, ID_A), MAC(K_2, (T_i\|\|C_{ab}\|\|B\|\| ENC(K_1, ID_A)))$	
Sends $\xrightarrow{D}$	Receives
Increments Counter, $C_{ab} = C_{ab} + 1$	

Phase IV: Identity Signature Verification

Node A	Node B



Checks bit B from data packet D.

Reads & decides T_i from local clock.

Calculates T_{delay}.

Checks if $T_{delay} > \Delta T - 2. \Delta t$, then proceed further otherwise terminates session.

If $T_{delay} <= \Delta T - 2$ then checks MAC value i.e. calculates MAC(K_2, ($T_i||C_{ab}||B||$ ENC(K_1, ID_A)))

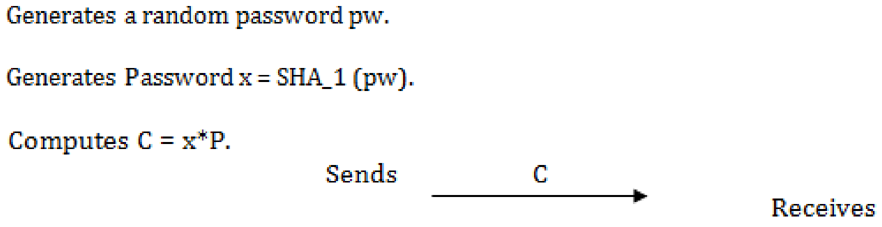
If calculated MAC==received MAC then Accept D otherwise Discard D.

Retrieves ID_A from DEC(K_1, ENC(K_1, ID_A)).

Increments Counter, $C_{ba} = C_{ba} + 1$.

Phase V: Password Generation

Node A	Node B



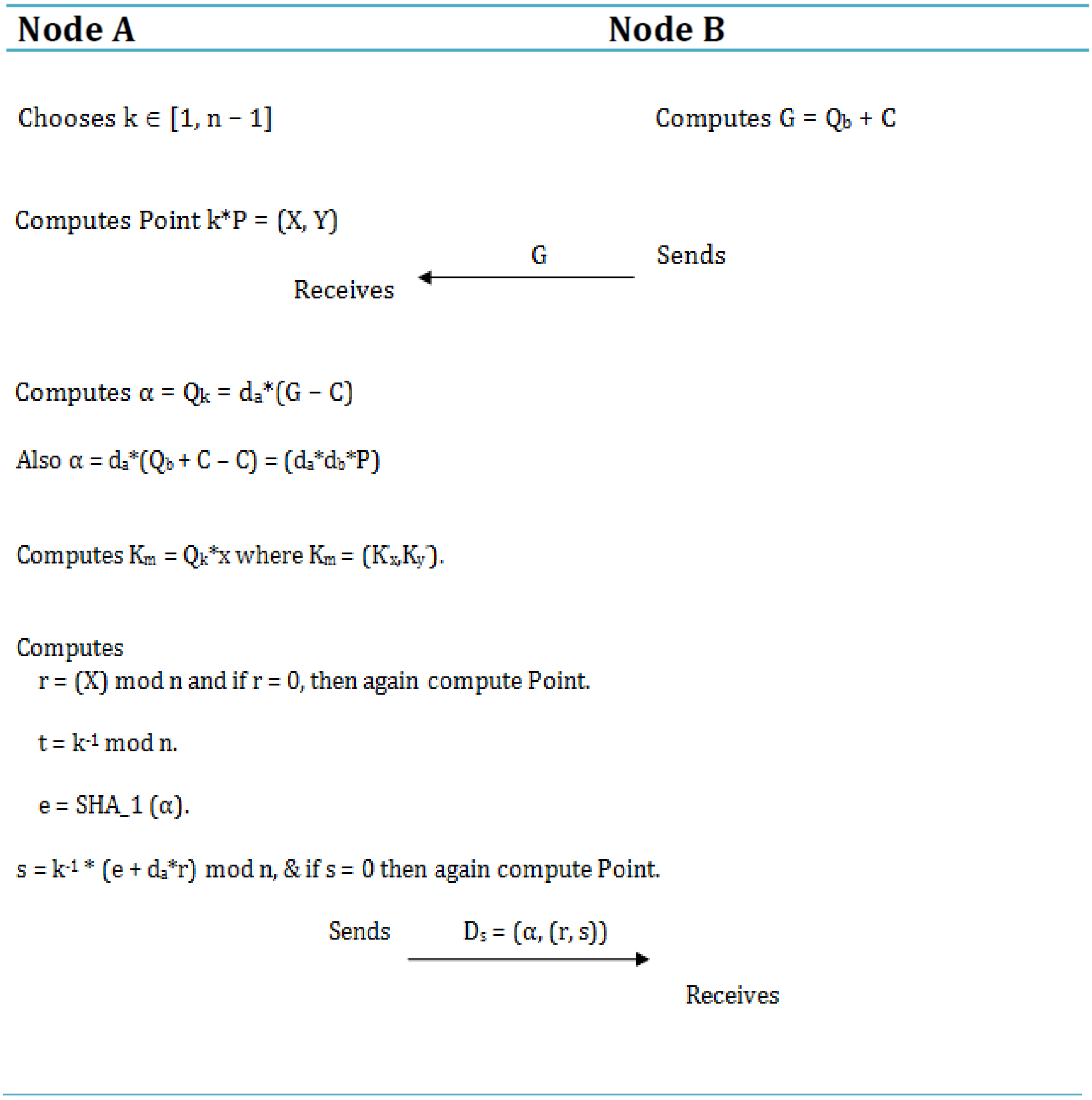
Generates a random password pw.

Generates Password x = SHA_1 (pw).

Computes C = x*P.

Sends $\xrightarrow{C}$ Receives

Phase VI: Node Signature Generation



Node A	Node B
Chooses $k \in [1, n - 1]$	Computes $G = Q_b + C$
Computes Point $k*P = (X, Y)$	

Receives $\xleftarrow{G}$ Sends

Computes $\alpha = Q_k = d_a*(G - C)$

Also $\alpha = d_a*(Q_b + C - C) = (d_a*d_b*P)$

Computes $K_m = Q_k*x$ where $K_m = (K_x, K_y)$.

Computes

$r = (X) \bmod n$ and if $r = 0$, then again compute Point.

$t = k^{-1} \bmod n$.

$e = SHA\_1 (\alpha)$.

$s = k^{-1} * (e + d_a*r) \bmod n$, & if $s = 0$ then again compute Point.

Sends $\xrightarrow{D_s = (\alpha, (r, s))}$ Receives

Phase VII: Node Signature Verification

Node A	Node B
	Computes $\beta = Q_k = d_b{*}Q_a = (d_a{*}d_b{*}P)$
	Generates $K_m = Q_k{*}\,x$.
	Computes $w = s^{-1} \bmod n$, $u_1 = (SHA\_1(\beta) * w) \bmod n$, $u_2 = (X * w) \bmod n$, $X = u_1{*}P + u_2{*}\,Q_a = (X_0, Y_0)$.
	Checks If $X_0 = O$ then reject the signature else compute v where $v = (X_0) \bmod n$.
	If ($v == r$) then Node B authenticates the Node A. Otherwise terminates session.
	Compute $Y_B = SHA\_1(\beta)$.
Receives $\xleftarrow{Y_B}$	Sends
Computes $Y_A = SHA\_1(\alpha)$	
If ($Y_A = = Y_B$) then Node A authenticates Node Otherwise terminates session.	

Phase VIII: Common Session Key Generation

Node A	Node B
Generates Common Session Key K_S where $K_S = (SHA\_1\ (ID_A \| \| ID_B \| \| T_i)*P + Q_k)$.	Generates Common Session Key K_S where $K_S = (SHA\_1\ (ID_A \| \| ID_B \| \| T_i)*P + Q_k)$.
Computes K_1' and K_2' where $K_1' = KDF\ (K''_x)$ $K_2' = KDF\ (K''_y)$ and $K_S = (K''_x, K_y'')$	Computes K_1' and K_2' where $K_1' = KDF\ (K''_x)$ $K_2' = KDF\ (K''_y)$ and $K_S = (K''_x, K_y'')$

Phase IX: Elliptic Curve Data Encryption Scheme

Node A	Node B
Creates Cipher Text c from Message M i.e. $c = ENC\ (K_1', M)$.	
Computes MAC value t of c i.e. $t = MAC\ (K_2', c)$.	
Sends $\xrightarrow{d=(c,t)}$	Receives

Phase X: Elliptic Curve Data Authentication Scheme

Node A	Node B
	Computes t' where $t' = MAC(K_2', c)$ If Computed MAC == Receives MAC value then data d is Accepted otherwise Discarded. If Accepted then decrypts Message M i.e. $M = DEC(K_1', c)$. Finally : Checks the current time T_C from local clock, T_0 the starting time of session & T_S, the session time elapsed during the whole communication. Compares If ($T_C - T_0 <= T_S$) If Comparison is True then session is Accepted. If Comparison is False then session is Discarded. **: Session Terminates**

Algorithm 4.1 Proposed ECC-Timestamp based Mutual Authentication & Key Management Scheme for Wireless Sensor Networks.

4.2.3 Explanation of Proposed Scheme:

Phase I: Mutual Agreed Key Generation & Exchange

Step 1: Node A takes a base point P and generates a random number k_a.

Step 2: Node A computes point $Q_a = k_a.P$ and sends Q_a to Node B.

Step 3: Node B generates a random number k_b and computes the point $Q_b = k_b.P$ and sends Q_b to the Node A.

Step 4: Node A now computes $P_1 = k_a.\ Q_b = (X_a, Y_a)$ and Node B computes $P_2 = k_b.Q_a\ = (X_b, Y_b)$.

Step 5: $K_{ab} = P_1 = P_2 = k_a.k_b.P$, where K_{ab} is used as the secret mutual agreed key and $K_{ab} = (K_x,K_y)$ is a base point representation on the elliptic curve E (F_2^m).

Step 6: Node A and Node B simultaneously checks that $P_1=P_2\neq O$ and computes $K_1=KDF(K_x)$ and $K_2=KDF(K_y)$ where KDF( ) is Key Distribution Function.

Note: From now onwards k_a is the private key and Q_a is the public key of Node A. Similarly k_b is the private key and Q_b is the public key of Node B and these keys will be further use in node signature and verification phases.

Checkpoint: It should be noted that the public key generated in the subsequent steps at both the ends needs to be validated in order to ensure that it satisfies the arithmetic requirement of the elliptic curve public key.

A public key $K = (K_x,K_y)$ associated with the domain parameters $\mathbf{D = (q, F_2^m, a, b, P, n, h, T_S)}$ is validated using the following procedure:

1. Check that $K \neq \boldsymbol{O}$.
2. Check that K_x and K_y are properly represented elements of F_2^m.
3. Check if K lies on the elliptic curve defined by a and b.
4. Check that $n.K = \boldsymbol{O}$.

Phase II: Time Stamp Generation

Step 1: Node A chooses the value of bit B to be "0" or "1" in order to denote even or odd data packet.

Step 2: Node A then chooses the value of T_i by reading current time from its local clock and then decides its corresponding numerical value where $T_i = T_0 + i.\Delta T$. Here T_i stands for Current Time Stamp, T_0 stands for starting time of time interval of the corresponding session established between participants viz. Node A and Node B, ΔT is a time interval of this particular session and i is a variable whose domain consists of integers i.e. $i \in \{-\varpi.....-2, -1, 0, 1, 2,......+\varpi\}$.

Assumption: We assume that base station has already securely distributed the unique node IDs to the different nodes before the establishment of session between any two corresponding participating nodes. In this scenario the nodes A & B have already received their unique node IDs from the base station and are able to exchange them in the next phase in a secure and efficient manner. Normally these unique IDs are not shared with the other sensor nodes of the same network until a request for session establishment is not accepted with the proper session timings corresponding to the local clock of the participating nodes.

Phase III: Identity Signature Generation & Exchange

Step 1: Node A generates the data packet D where

$D = (B \,||\, E_{K1}(ID_A), MAC_{K2}(T_i||C_{ab}||B||\, E_{K1}(ID_A))$

Step 2: Node A sends data packet D to the Node B.

Step 3: Node A increments its corresponding counter value i.e.

$C_{ab} = C_{ab} + 1$.

Phase IV: Identity Signature Verification

Step 1: Node B checks the bit B's value from data packet D.

Step 2: Node B now reads the current time from its local clock and then decide the value of its current time stamp T_B based on it and also confirms the "0" or "1" bit value of B and finally calculates the time delay T_{delay} for that.

Step 3: Node B checks the condition, that if $T_{delay} > \Delta T – 2. \Delta t$ for a receiving packet from Node A then that packet is considered to be from another or different time interval and therefore that packet is discarded.

Step 4: If $T_{delay} <= \Delta T – 2. \Delta t$ for a receiving data packet then Node B checks the MAC value of that packet by calculating $MAC_{K2}(T_i||C_{ab}||B|| E_{K1}(ID_A)$.

Step 5: If the computed MAC value at Node B's side doesn't match the received MAC value then that packet is discarded otherwise its accepted and ID_A is derived by decrypting the encrypted portion of the message i.e. $D_{K1}\{E_{K1} (ID_A)\}$.

Step 6: Node B now increments its corresponding counter value i.e. $C_{ba} = C_{ba} + 1$.

Explanation: Consider a situation where the two local clocks of the corresponding participating nodes A & B are not accurately synchronized but they are rather loosely synchronized.

Let us assume that the maximum difference between A's & B's clock is Δt or it's the relative error between the two local clocks, as shown in Fig.3 and Fig.4. Now assume that Node B received a packet in time interval $(T_0 - \Delta t, T_0)$. If the packet is labelled as a "1" packet then it's obvious that it came from $(T_0 - \Delta T, T_0)$ time interval and if that is labelled as a "0" packet then it might came from $(T_0,T_0+\Delta T)$ time interval that too if Node B's local clock is slower than Node A's or it might came from the time interval $(T_0-2*\Delta T, T_0-\Delta T)$ if Node B's local clock is faster than Node A's. From this we figure out that the time interval at which the packet has arrived still satisfies the condition for delay i.e. $T_{delay} < \Delta T$.

Now as a matter of fact this data packet may have arrived from the time interval $(T_0, T_0+\Delta T)$ or $(T_0-2*\Delta T, T_0-\Delta T)$ since we are unaware of the fact that whether Node B's local clock is faster or slower than Node A's. However if the corresponding packet is from time interval $(T_0-2*\Delta T, T_0-\Delta T)$, the minimum delay is the time elapsed between the transmitting time interval i.e. $(T_0- \Delta T- \delta)$ where δ is a very small number tending to the value zero, according to Node A's clock and received at time interval $(T_0 - \Delta t + \delta)$ according to Node B's local clock. The worst case can be when Node B's local clock is faster than A's. Now according to Node A's local clock the packet is received at time interval $(T_0 - \Delta t + \delta - \Delta t) = (T_0 – 2*\Delta t + \delta)$. Then, the delay is computed as:

$$T_{delay} = (T_0 – 2*\Delta t + \delta) - (T_0 – \Delta t - \delta) = (\Delta T - 2*\Delta t + 2*\delta).$$

Now if the data packet is received at the time interval (T_0, $T_0 + \Delta T - \Delta t$) at Node B, then "0" packet comes from time interval (T_0, $T_0+\Delta T$) and "1" packet comes from the time interval ($T_0 - \Delta T$, T_0). So finally if $T_{delay} < (\Delta T - 2*\Delta t)$, one can decide that from which time interval the respective data packet came or arrived from. As a result of this, if a data packet's delay is larger than ($\Delta T - 2*\Delta t$), it will be considered as a data packet from another time interval. As a consequence of which a wrong T_i will be used by the Node B to calculate the MAC value and as a result of which the recently computed MAC value will not match the received MAC value. Finally due to the mismatch the data packet is discarded [12].

Note: In this manner Node B retrieves the unique node ID of Node A. In the similar fashion Node B chooses the value of bit B and correspondingly decides the value of T_i and in turn generates the same data packet with its unique node ID and sends it along with the corresponding MAC. After receiving the data packet D from Node B, Node A follows the same procedural steps in order to validate Node B's signature and at the end Node A also retrieves the unique node ID of Node B.

Phase V: Password Generation

Step 1: Node A chooses a random password pw and computes x = SHA_1 (pw).
Step 2: Further, Node A computes C = x*P.
Step 3: Lastly Node A announces C to Node B in a secure manner.

Phase VI: Node Signature Generation

Step 1: Node A randomly chooses a number $k \in [1, n - 1]$ and computes Point k*P = (X, Y).
Step 2: Node B simultaneously computes $G = Q_b + C$ and sends G to the Node A.
Step 3: Node A receives G and computes $\alpha = Q_k = d_a*(G - C) = d_a*(Q_b + C - C) = (d_a*d_b*P)$
Step 4: Node A now generates K_m, a secret mutual agreed key shared between the participating nodes A and B in this session for the purpose of authenticating each other mutually where $K_m=(K'_x,K_y')$ is a base point representation on the elliptic curve E (F_2^m) and is computed as $K_m = Q_k*x$.
Step 5: Node A computes three modular arithmetic parameters viz. r = (X) mod n and if r = 0, then goto Step 1.

$t = k^{-1} \bmod n.$

$e = SHA\_1\ (\alpha).$

$s = k^{-1} * (e + d_a*r) \bmod n$, & if s = 0 goto Step 1.

Node A now sends data packet $D_s = (\alpha, (r, s))$, where (r, s) is signature pair of Node A.

Phase VII: Node Signature Verification

Step 1: After receiving the data packet D_s Node B computes $\beta = Q_k = d_b*Q_a = (d_a*d_b*P)$ and finally generates secret mutual agreed key and in turn computes $K_m = Q_k* x$.

Step 2: Node B now computes the following modular arithmetic parameters viz.

$w = s^{-1} \bmod n.$

$u_1 = (SHA\_1(\beta) * w) \bmod n.$

$u_2 = (X * w) \bmod n.$

$X = u_1*P + u_2* Q_a = (X_0, Y_0).$

Step 3: Node B now performs the following computations viz.

If $X_0 = O$, then reject the signature else compute v, where $v = (X_0) \bmod n$.

If (v == r) then Node B authenticates the Node A and send Y_B where

$Y_B = SHA\_1(\beta)$ to Node A for authentication.

Step 4: Node A on receiving Y_B, computes Y_A for comparison purpose where

$Y_A =$ SHA_1(α) and if (Y_A = = Y_B) then Node A also authenticates Node B.

Proof for Verification: If the corresponding message is indeed signed by Node A, then $s = k^{-1} * (SHA\_1\ (\alpha) + d_a*r) \bmod n$ which means, $k = s^{-1} * (SHA\_1\ (\alpha) + d_a*r) \bmod n$

$k = (u_1 + u_2.d_a) \bmod n$.

Now consider $(u_1*P + u_2* Q_a) = (u_1 + u_2.d_a) *P = k*P$.

Further Point $X = u_1*P + u_2* Q_a = (X_0, Y_0)$.

Thus we see that v = r, since r = X mod n where X is the x coordinate of the point (k*P) and we have already shown that

$(u_1*P + u_2* Q_a) = k*P$. Hence its proved.

Phase VIII: Common Session Key Generation

Step 1: Finally Node A & Node B agree on a common session key K_S where

$K_S = (SHA\_1(ID_A||ID_B||T_i)*P + Q_k)$ and generates K_S at both the ends respectively.

Step 2: Node A and Node B respectively computes $K_1^{'}$ and $K_2^{'}$

where $K_S = (K^{''}_x, K_y^{''})$

$K_1^{'} = KDF\ (K^{''}_x)$

$K_2^{'} = KDF\ (K^{''}_y)$

Phase IX: Elliptic Curve Data Encryption Scheme

Step 1: Node A finally creates the cipher text c using the Message Block M, where $c = ENC(K_1^{'}, M)$.

Step 2: After the creation of cipher text c, Node A computes the MAC value of the cipher text i.e. $t = MAC(K_2^{'}, c)$.

Step 3: Node A sends data packet d = (c, t) to the corresponding participating Node B in a secured manner through communication link established for this particular session.

Phase X: Elliptic Curve Data Authentication Scheme

Step 1: After receiving data packet d, Node B computes $t^{'}$ where $t^{'} = MAC(K_2^{'}, c)$. Now if the computed MAC value matches the received MAC value of the cipher text then the data packet is verified and finally Node B decrypts the corresponding Message Block M i.e. $M = DEC(K_1^{'}, c)$.

Note: In this way through a secure communication channel established for a particular session, the critical information from one sensor node is transmitted and shared with another. This is achieved with reasonable security and with less processing overhead and that too with little memory utilization.

Lastly the Sensor Node at the receiving end i.e. Node B checks the current time T_C from its local clock and also the Time Stamp T_0 where T_0 is the starting time of the session noted earlier & calculates the total session time by comparing it with the Time Stamp T_0 and if the difference is found to be less than the threshold time interval T_S i.e. if $T_C - T_0 <= T_S$ then that session is termed as a valid one and the received message block in the form of data packet is accepted otherwise that established session is terminated and the corresponding communication link is discarded along with corresponding data packet.

CHAPTER 5: SECURITY PERSPECTIVE AND ANALYSIS OF PROPOSED SCHEME

5.1 Analysis of Attacks on Wireless Sensor Nodes:

In this section, we discuss the security of the proposed scheme. The proposed ECC-Time Stamp based mutual authentication and key management scheme will be considered to be a secure authenticated key establishment scheme, if it can resist the following attacks on WSNs as mentioned in [9]:

- **Passive Information Gathering:** In this class of attack an attacker can attempt to pick on a data stream containing important information and can also attempt to listen the data being communicated among various sensor nodes in the domain of WSNs. If the attacker belongs to the class of a laptop class attacker, he can very easily extract physical locations of sensor nodes and can easily attempt to destroy them. The main motive behind destroying the sensor nodes may be to make the network dysfunctional or dividing a large single network of WSNs into two smaller networks.

  In order to counter this type of attack we had proposed to use a strong ECC based Encryption and Decryption schemes in the proposed work as we know that the corresponding arithmetic operations on the elliptic curve points exploits the security factor associated with the elliptic curve equation defined over a finite field and the difficulty associated with finding the set of solutions in the domain of ECC arithmetic. In the proposed scheme we have used a private key, $K_1^{'}$ that has been derived by applying key derivation function on the subsequent common session key, K_S which ensures security for the privately encrypted message block. Also in order to maintain the authenticity of the encrypted message block we had proposed to use the Message Authentication Code for verification purpose. Thus our proposed scheme is secure against Passive Information Gathering Attack.

- **Sleep Deprivation Attack:** The basic concept behind this attack is to keep the corresponding sensor node which had been deployed in a hostile environment, deprived from its energy saving sleep mode, thereby reducing its power resources. This kind of attack is a very intelligent type of attack because in this scenario the corresponding attacker can make the requests seem to be legitimate, resulting in the sensor death. The attack exploits the fact that wireless sensors networks may be deployed in areas where the environment is hostile and the nodes cannot be serviced frequently.

  To counter such an attack we had proposed to use a strong ECC based authentication techniques for the mutual authentication of the corresponding participating nodes in a session so that we can prevent the attacker from pretending to be a valid sensor in the network. Also the mutual authentication of the corresponding participants' viz. Node A & Node B in a session is performed twice firstly using the current time stamp T_i and a unique time stamping mechanism in Phase IV and secondly using one way hash function SHA_1( ) on the unique IDs of the participating sensor nodes for the generation of unique password at both the ends and then using that unique password for corresponding sensor nodes mutual authentication in Phase VII. In this manner the proposed scheme resists the Sleep Deprivation Attack and is secure against it.

- **Sinkhole Attack:** In this type of attack the main motive of a laptop class attacker or a simple attacker is to direct all network traffic towards the self deployed sensor node in the corresponding hostile environment for the purpose of intercepting or hacking the respective communication channel between the corresponding sensor nodes participating in a secure session for information exchange. The attack is usually initiated by attacking a particular sensor node deployed in a hostile environment and then making it a better choice for routing. All neighbouring sensor nodes are given the misconception that traffic passing through the compromised node adopts an efficient route. A successful attack results in the creation of a large "sphere of influence" in which traffic originating from nodes several hops away is routed through the compromised node and then towards the intended base station or the self deployed sensor node.

The sinkhole attack can be defeated if the communicating nodes in a particular session use a strong verification mechanism that verifies the quality and correctness of the node identities in each and every established session efficiently. For the purpose of introducing strong verification of the corresponding sensor nodes in a session we had proposed to use an ECC based sensor node identity verification scheme in Phase VII that too with the unique ID and password generation and verification scheme. The unique password generated in Phase V ensures that for each and every new session a new random password is created with the help of Sensor Node IDs and the current Time Stamp T_i where password is computed as $x = SHA\_1 (ID_A||ID_B||T_i)$ which in turn ensures perfect secrecy and security during the mutual verification of corresponding sensor nodes participating in a particular session.

This attack has also been resisted by adopting a strong ECC based authentication protocols in the proposed work. As it is very much clear from the Phase IV and Phase VII in the proposed work that the mutual authentication of the corresponding sensor nodes had been performed twice firstly with the use of current Time Stamp T_i and secondly with the use of unique password which in turn had been derived from the unique IDs of the sensor node along with T_i. The main motive behind double mutual authentication and verification is to make the proposed scheme resistant towards Sinkhole Attack and also to nullify the drawbacks of loosely synchronised local clocks in Phase IV. Thus we can now state that our proposed scheme is very much secure against the Sinkhole Attack and it can efficiently resist this attack.

- **Wormhole Attack:** In this type of attack usually the laptop class attackers tend to coordinate with each other in an efficient manner to use a low latency side channel for communication. Message blocks are sent from one attacker to the other using this side channel. This gives other nodes the perception that the attacker is a better node for routing. Another effect of this attack is that while the side channel exists communications are enhanced instead of being denied. When the attacker leaves, the network loses its enhanced capacity and may require re-initialization of network services to increase throughput.

  The mechanism employed in the proposed scheme for defeating wormhole attack is the Time Stamping Mechanism which has been used in Phases II, III, and IV.

In the proposed Time Stamping Mechanism a current Time Stamp T_i along with a bit B ensures that the transmission range of the sender sensor node has not exceeded and if it exceeds then the authentication process employed at receiver’s end ensures that the corresponding received data packet along with the established communication channel is discarded and the session is terminated immediately. The mechanism is capable of handling loosely synchronised local clocks of the participating Sensor Nodes and is therefore more useful in case of wireless sensor networks. In this manner the proposed scheme with the time stamping mechanism resists wormhole attack in a safe and secure way.

- **Sybil Attack:** Mostly the protocols related to security and networks require that a network entity must have a single unique identification. For instance in a wireless sensor network every sensor node must have a single unique identification. In this attack the adversary can possess multiple identities. If these multiple identities are united with fake locations then the attacker can appear to be present at large number of locations with different identities. If an attacker launches an attack in the vicinity of a victim then it will appear at large neighbouring locations around the victim. This increases the probability of the attacker being chosen for geographic forwarding in the next hop.

  The proposed scheme with the help of strong ECC based mutual authentication and verification processes ensures that in each and every newly established session the identity and vicinity of the participating nodes are properly authenticated and verified respectively. Further the time mechanism ensures that each and every sensor node’s location is indirectly verified with the help of chosen Current Time Stamp T_i and associated communication delay T_{delay} in transmission of data packet. In this way the proposed scheme resists the Sybil Attack that too in a safe and secure manner.

- **HELLO Flood Attack:** In this attack the main aim of the attacker is to announce itself to be the neighbour of another node. Many protocols require that “HELLO” messages need to be constantly exchanged among the sensor nodes to establish neighbourhood tables. These neighbourhood tables can also affect the routing tables resulting in network traffic diversions. This attack is more successful if launched by a laptop class intruder with large enough transmission power to effectively broadcast to the entire network that the adversary is the actual neighbour. Even if a node is not able to exchange messages with the adversary it will be transmitting messages into ‘thin space’ thereby resulting in loss of network potential.

In order to counter the HELLO Flood Attack, the proposed scheme has been designed in such a way that it tends to verify the bidirectional behaviour of the corresponding communication link before the session establishment during the Phase III and Phase IV respectively. Also to tackle the same problem an ECC based secure authentication mechanism has already been included in the scheme which in turn controls the HELLO Flood Attack to a much greater extent. In this manner the proposed scheme counters HELLO Flood Attack in a safe and an efficient way.

- **Jamming Attack:** This attack is a type of denial of service (DoS) attack in which the major focus of the attacker is to jam communications between the corresponding sensor nodes in a session. This attack can be launched by simply transmitting a jamming signal that interferes with radio frequencies being used by the sensor network.

  The jamming attack is of two types: constant jamming and intermittent jamming. Constant jamming requires that the attacker completely jams the entire network and so there can be no exchange of messages among respective sensor nodes. Intermittent jamming allows sensors to exchange messages regularly but not at a consistently. Although the intermittent attack allows message exchange to a certain extent but the result can be equally as bad as the constant jamming. Choice of jam attack depends on the amount of resources available to the attacker. If the attacker is strongly equipped then he would resort to the use of the constant jam.

  In order to counter this attack, the threshold session time T_s along with the current and starting time stamp T_C and T_0 has been employed in the scheme after the completion of Phase X so that any subsection of the wireless sensor network deployed in a hostile environment doesn’t get jammed at any moment of time, as it has been mentioned in the proposal that if T_C - T_0 <= T_S then only session will be accepted as a valid one otherwise it will be discarded. The proposed scheme prevents the nodes beyond the defined vicinity to participate in a session establishment process with any of the other sensor nodes in a hostile network. In addition to this the proposed scheme doesn’t allow any established session to go beyond the threshold session time and in turn helps in managing traffic and prevents jamming in any section or subsection of the deployed sensor network in a dynamic environment. Thus we are now in a position to state that the proposed scheme also resists this type of Jamming Attack or rather prevents the jamming to a certain extent.

5.2 Analysis of General Attacks on any Participating Node:

In the subsequent section, we discuss the security factors associated with the proposed scheme, not restricting to the domain of wireless sensor networks but we tend to analyse the general security parameters associated with any secure mutual authentication and key management scheme in real world applications and at the same time we tend to discuss the most common attacks on any mutual authentication scheme and how the proposed scheme is secure against them [13]:

- **Dictionary attack**: The dictionary attack could be performed in offline or online mode. An on-line password guessing attack cannot succeed since sender Node A and receiver Node B can choose appropriate trail intervals during time stamp generation and exchange in Phase II and Phase III. On the other hand, in an off-line mode password guessing attack, it is impossible to get the real password since a one way hash function is applied to the password where password $x = SHA\_1(ID_A||ID_B||T_i)$. In the proposed scheme, the shared mutual agreed key Q_K used in the calculation of Ks is calculated from d_a and d_b which are freshly generated in every new session; and by assuming the intractability of Elliptic Curve Discrete Logarithm problem. Therefore the proposed scheme securely resists both the offline and online dictionary attack.

- **Passive attack**: A passive attack can be possible if Z, the attacker, makes a guess at the session key K_S using only the information which is obtainable over network in any domain. If the attacker Z performs a passive attack, then the proposed verification mechanism automatically terminates the session between both the accepting parties. That is, Node B and Node A successfully identifies each other and also efficiently authenticates each other, and they both in turn compute the session key K_S. So, Z the adversary, cannot compute any information about the common shared session key K_S due to the intractability of Elliptic Curve Discrete Logarithm problem. Therefore the proposed scheme resists or rather counters the passive attack.

- **Man in the middle attack**: It can be considered as an active attack. In this scheme no useful information about the secret mutual agreed key Q_K is revealed during a successful run. If an attacker Z intercepts G and replaces it with G', Z then receives α and (r, s) from Node A. C will try to replace s by s′, as before. However, this means that Z must calculates α where $\alpha = d_a^* (G' - C)$ but Z cannot compute the value of α because Z does not know the value of C neither the value of d_a. So, Z will not be able to compute Q_K, neither K_S. Thus this scheme securely resists the man in the middle attack.

- **Perfect forward secrecy**: In perfect forward secrecy, even if the Node's secret password is compromised, it never allows the adversary to determine the session key K_S for past sessions and decrypt them. In the proposed scheme, the work is based upon the assumption that the Elliptic Curve Discrete Logarithm problem is intractable and on the value of the key $K._{S.}$ Even if the attacker knew the correct password, the attacker still cannot compute the previous session keys because K_S is derived from the common mutual agreed key Q_K which is generated from the random values of d_a and d_b. Thus the property of perfect forward secrecy is satisfied by the proposed scheme.

- **Known-key attack**: In our proposed scheme, the sender Node A and the receiver Node B both generates new set of random numbers d_a and d_b and the corresponding secret mutual agreed key K_{ab} in Phase I and a different set of secret mutual agreed key K_m in Phase VI and is generated with every new session. Another important aspect of the proposed protocol is that the secret common session key K_S generated in Phase VIII, is calculated independently on both sides and protected by the secure hash function. Thus the proposed scheme is secure against the known key attacks assuming that the Elliptic Curve Discrete Logarithm problem is intractable.

CHAPTER 6: IMPLEMENTATION DETAILS AND PERFORMANCE ANALYSIS

6.1 Implementation Details:

In implementing ECC based cryptographic modular phases in form of real time domain specific applications, generally two families of Elliptic Curves are used viz. Prime Curves defined over GF(p) and Binary Curves defined over GF(2^k).

In our proposed work, the scheme based on ECC arithmetic uses GF(2^k) as it offers significant advantages in performance over GF(p) in the domain of wireless sensor networks where constrained memory and limited processing power are the key issues which needs to be taken into consideration while proposing or implementing any mutual authentication and key management scheme. A Koblitz Curve over GF(2^{163}) was selected and used in implementing the proposed scheme. The focus of the implementation was to minimize execution time, while targeting the domain specific characteristics of WSNs, by maintaining an acceptable code size and minimizing memory and power consumption. Optimal Finite Field and Elliptic Curve parameters were sought for implementation. Any attempt to propose a secure and computationally efficient scheme for WSNs need to justify its computational cryptographic cost and the corresponding computational performance in terms of efficient processing time along with minimal utilization of memory with minimum processing overhead.

In order to evaluate the proposed scheme on the scale of cryptographic cost and computational performance, it was implemented using Language C in characteristic phases individually whereas the mutual authentication and signature verification phases were implemented using Java SE 6. The basic functionality for using cryptographic techniques in Java is provided by the Java Cryptography Architecture (JCA) and its sibling, the Java Cryptography Extension (JCE). Application code is written that calls the appropriate

JCE/JCA API classes; these in turn invoke the classes in a provider that provides implementations for the JCE/JCA service provider interface (SPI) classes. The classes then invoke the internal code in the provider to provide the actual functionality requested.

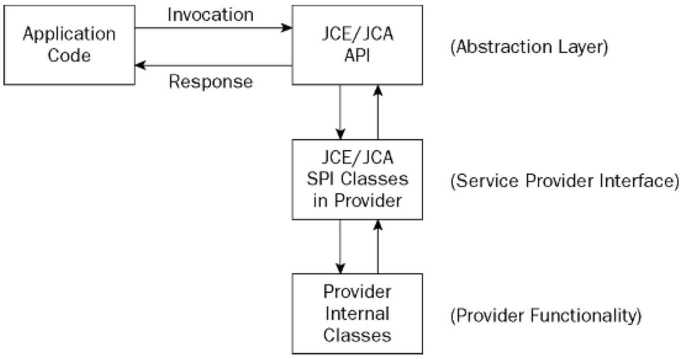


Figure 6.1: Interaction between Classes of JCE/JCA within a Cryptographic application developed using Java SE 6.

6.2 Software Architecture of ECDSA:

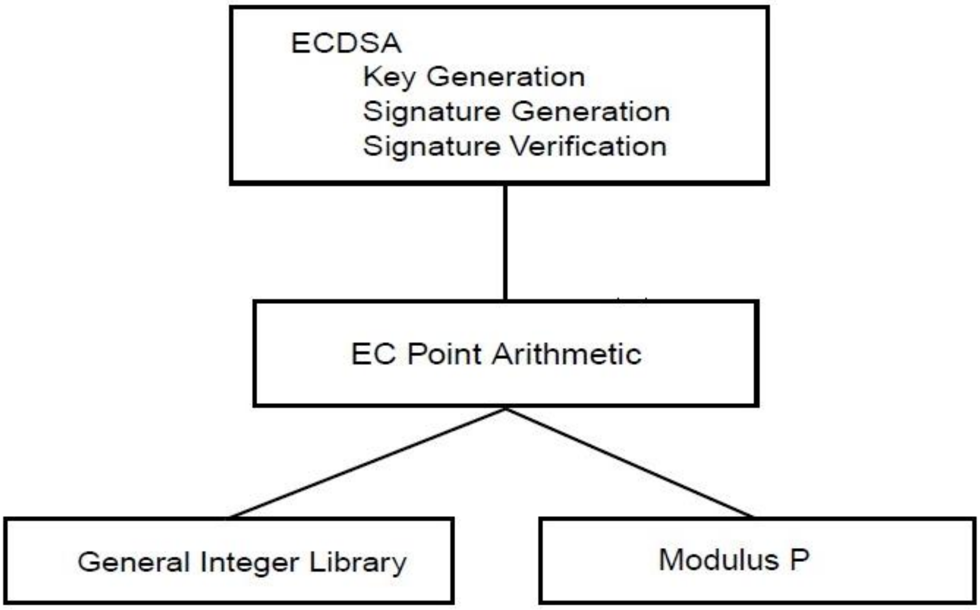


Figure 6.2: Software Architecture of Characteristic Phase ECDSA in Proposed Scheme.

6.3 Class Structure of Implemented Modules:

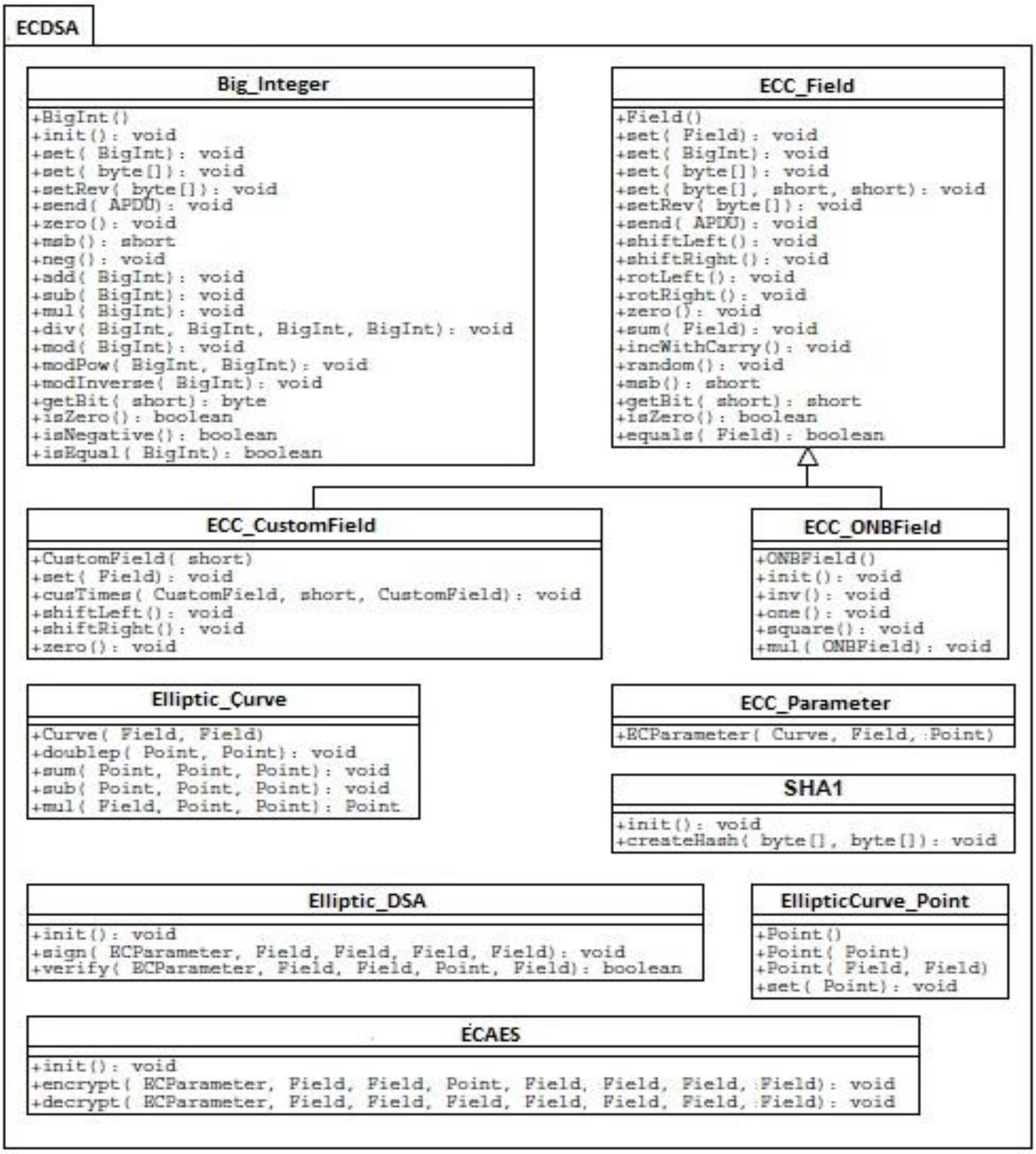


Figure 6.3: Class Structure of ECDSA Implementation in Proposed Scheme.

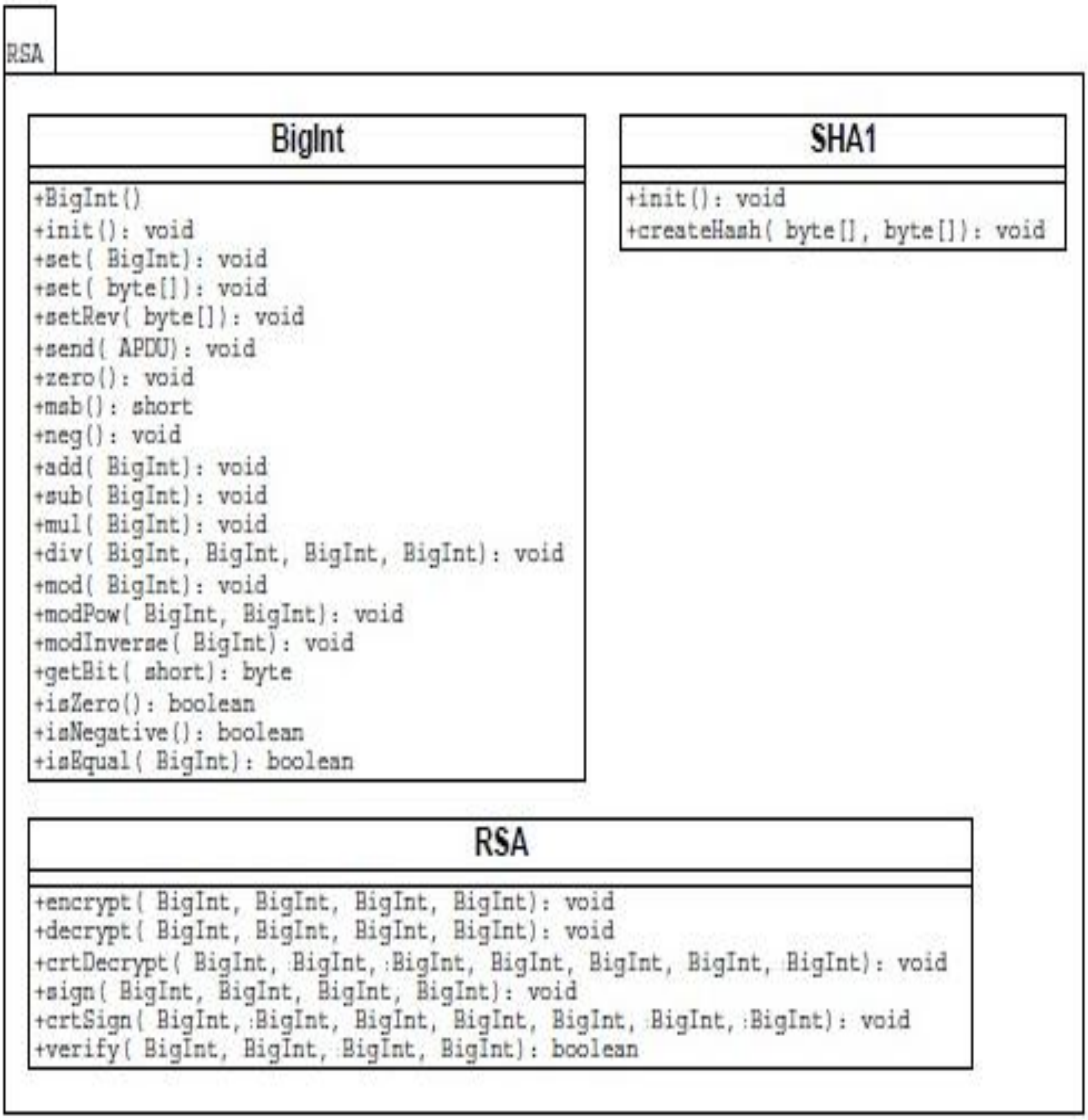
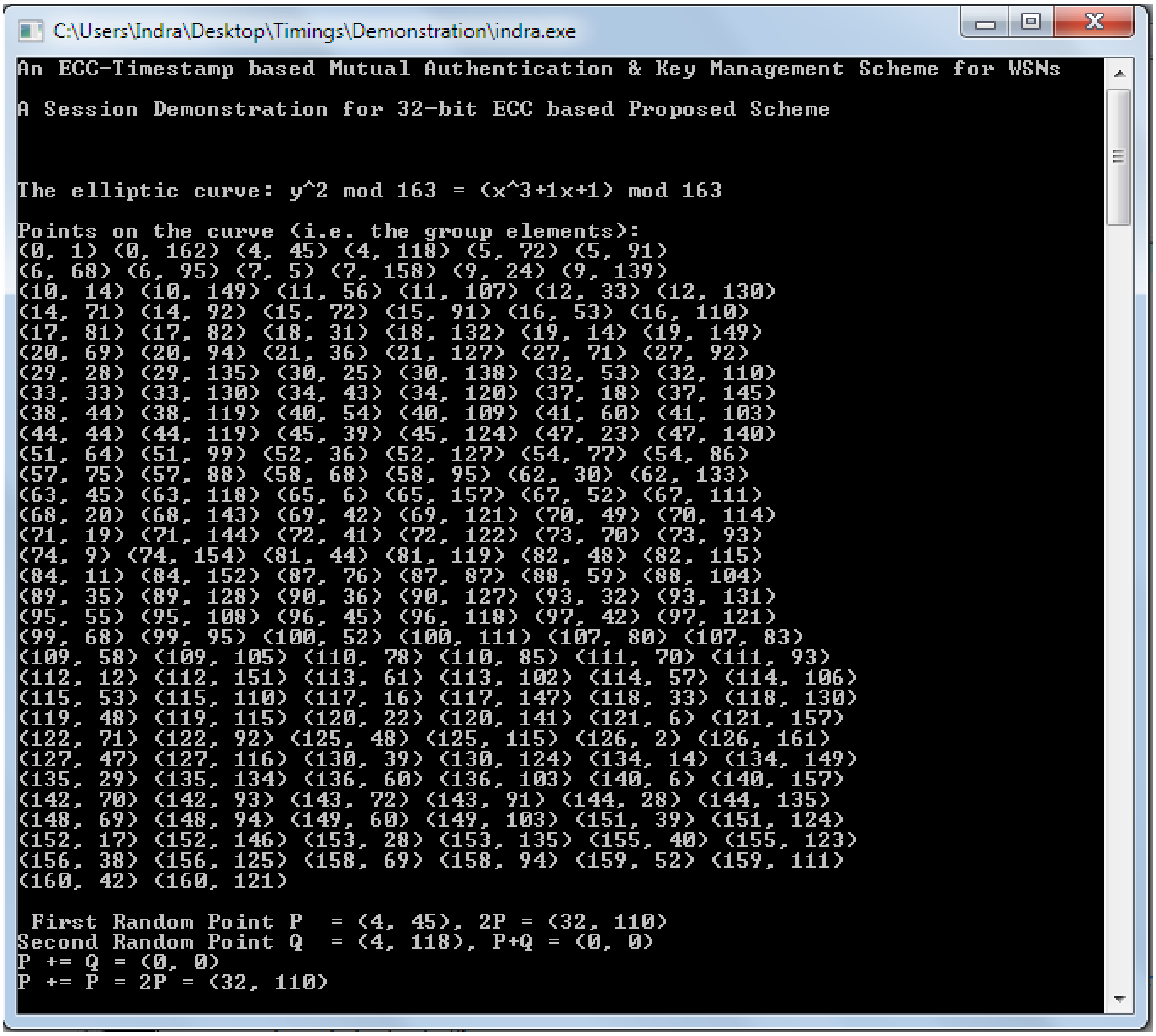

RSA
BigInt
+BigInt()
+init(): void
+set(BigInt): void
+set(byte[]): void
+setRev(byte[]): void
+send(APDU): void
+zero(): void
+msb(): short
+neg(): void
+add(BigInt): void
+sub(BigInt): void
+mul(BigInt): void
+div(BigInt, BigInt, BigInt, BigInt): void
+mod(BigInt): void
+modPow(BigInt, BigInt): void
+modInverse(BigInt): void
+getBit(short): byte
+isZero(): boolean
+isNegative(): boolean
+isEqual(BigInt): boolean
SHA1
+init(): void
+createHash(byte[], byte[]): void
RSA
+encrypt(BigInt, BigInt, BigInt, BigInt): void
+decrypt(BigInt, BigInt, BigInt, BigInt): void
+crtDecrypt(BigInt, BigInt, BigInt, BigInt, BigInt, BigInt, BigInt): void
+sign(BigInt, BigInt, BigInt, BigInt): void
+crtSign(BigInt, BigInt, BigInt, BigInt, BigInt, BigInt, BigInt): void
+verify(BigInt, BigInt, BigInt, BigInt): boolean

Figure 6.4: Class Structure of RSA Implementation.

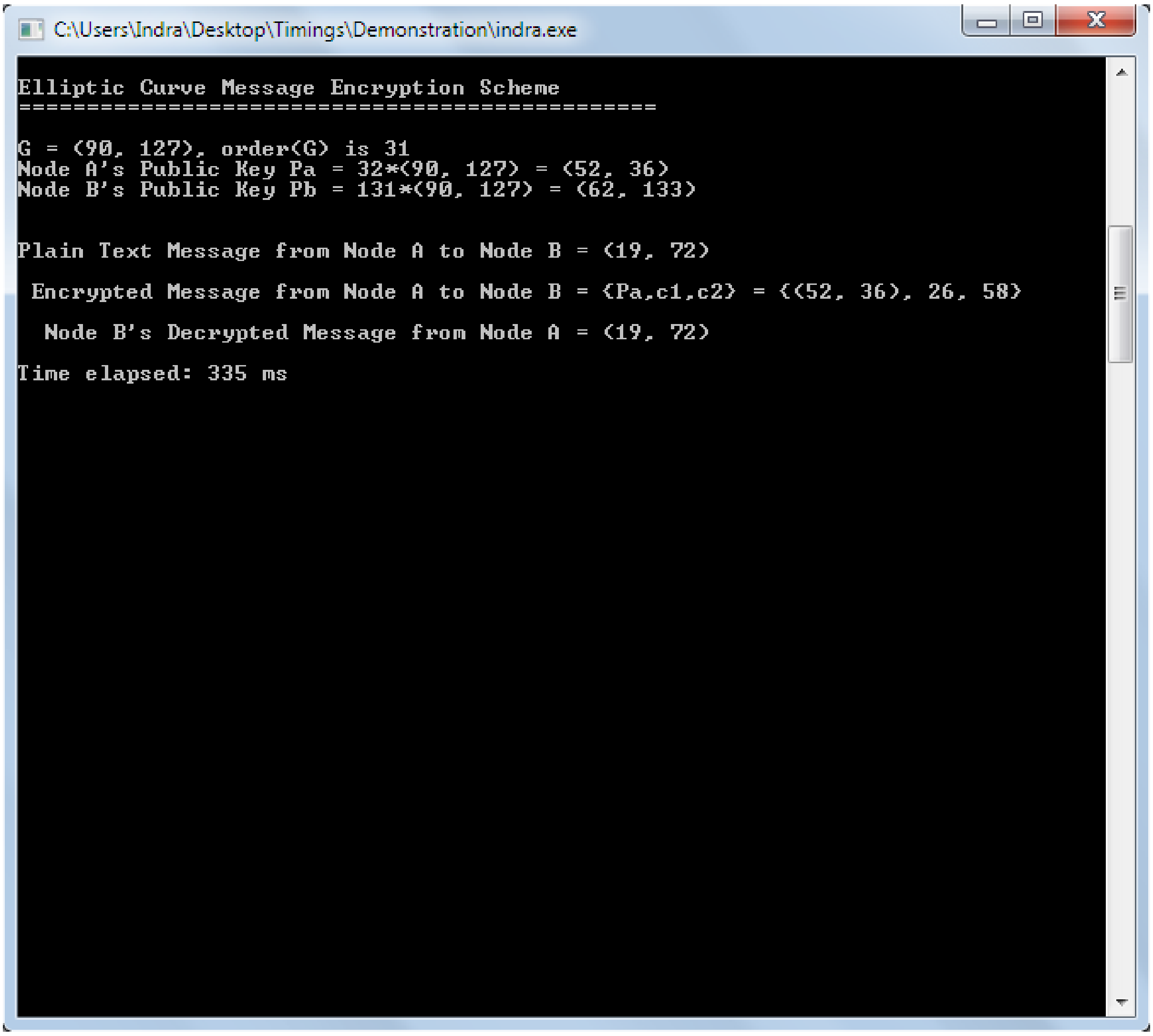
6.4 Results and Discussion:

The proposed scheme has been mainly implemented for measuring performance of 163-bit ECC-Timestamp based Mutual Authentication and Key Management Scheme and the estimated cryptographic computational cost for an established session between the two corresponding Nodes A & B is found to be 6408.1 ms for a single session as compared to 352302 ms for 1024-bit RSA Implementation (e=3) of the same scheme which in turn have been calculated by the summation of the individual phase timings of the proposed scheme. The findings of [Wang and Li 2006] in [25] states that 160-bit ECC signature generation and signature verification on MICA mote with ATMEGA128L CPU takes 1.3 sec and 2.8 sec respectively. The findings of the proposed scheme in this dissertation are found to be optimal in comparison to the findings of [Wang and Li 2006] in [25]. These statistics reveal that our proposed scheme manages to reduce the processing time of the cryptographic loads associated with the mutual authentication and key management in the specific domain of wireless sensor networks. The simulation or implementation of the proposed scheme for the respective n-bit ECC-Time Stamp based protocol and corresponding 1024-bit RSA-Time Stamp based protocol has been performed on the 2.13 GHz Intel Core i3 CPU installed with 1 GB of RAM running Windows 7 and the performance timings have been recorded in milliseconds (ms). The computational performance of this proposed work relies directly on the efficiency of asymmetric public key cryptographic techniques employed. Apart from the comparison of 163-bit ECC Time Stamp based proposed scheme with the 1024-bit RSA based proposed scheme, the performance timings of 176-bit, 192-bit, 208-bit and 256-bit ECC Time Stamp based proposed schemes have also been recorded for mutual comparison and statistical analysis. All these detailed statistical analysis are summarized as follows:



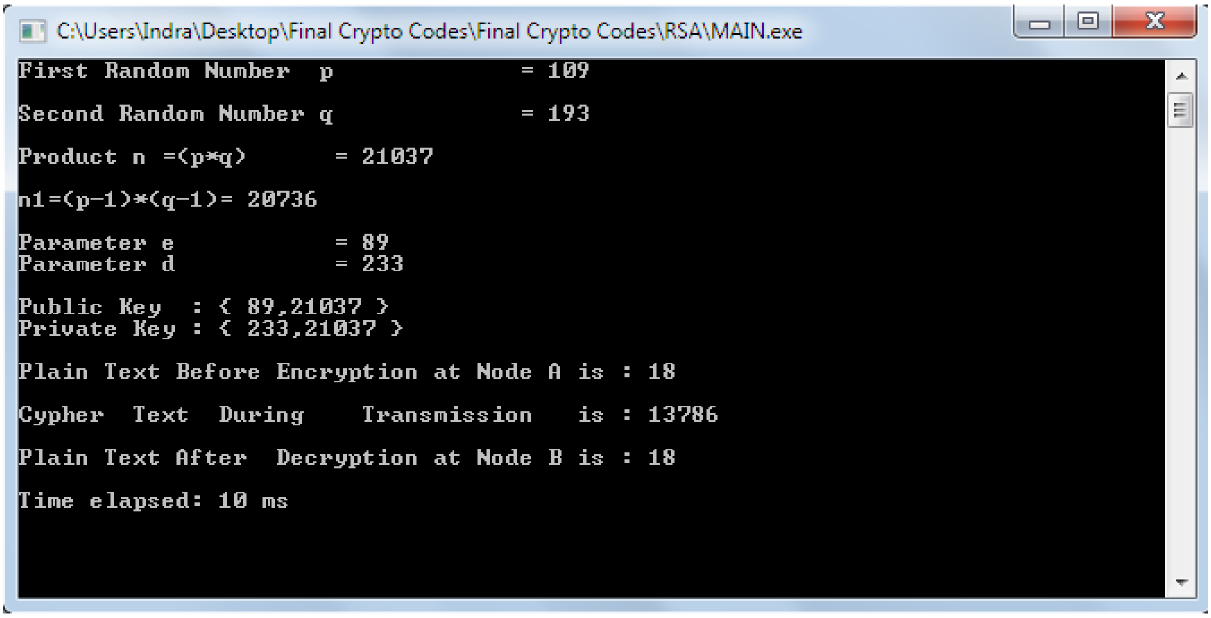
```
An ECC-Timestamp based Mutual Authentication & Key Management Scheme for WSNs

A Session Demonstration for 32-bit ECC based Proposed Scheme



The elliptic curve: y^2 mod 163 = (x^3+1x+1) mod 163

Points on the curve (i.e. the group elements):
(0, 1) (0, 162) (4, 45) (4, 118) (5, 72) (5, 91)
(6, 68) (6, 95) (7, 5) (7, 158) (9, 24) (9, 139)
(10, 14) (10, 149) (11, 56) (11, 107) (12, 33) (12, 130)
(14, 71) (14, 92) (15, 72) (15, 91) (16, 53) (16, 110)
(17, 81) (17, 82) (18, 31) (18, 132) (19, 14) (19, 149)
(20, 69) (20, 94) (21, 36) (21, 127) (27, 71) (27, 92)
(29, 28) (29, 135) (30, 25) (30, 138) (32, 53) (32, 110)
(33, 33) (33, 130) (34, 43) (34, 120) (37, 18) (37, 145)
(38, 44) (38, 119) (40, 54) (40, 109) (41, 60) (41, 103)
(44, 44) (44, 119) (45, 39) (45, 124) (47, 23) (47, 140)
(51, 64) (51, 99) (52, 36) (52, 127) (54, 77) (54, 86)
(57, 75) (57, 88) (58, 68) (58, 95) (62, 30) (62, 133)
(63, 45) (63, 118) (65, 6) (65, 157) (67, 52) (67, 111)
(68, 20) (68, 143) (69, 42) (69, 121) (70, 49) (70, 114)
(71, 19) (71, 144) (72, 41) (72, 122) (73, 70) (73, 93)
(74, 9) (74, 154) (81, 44) (81, 119) (82, 48) (82, 115)
(84, 11) (84, 152) (87, 76) (87, 87) (88, 59) (88, 104)
(89, 35) (89, 128) (90, 36) (90, 127) (93, 32) (93, 131)
(95, 55) (95, 108) (96, 45) (96, 118) (97, 42) (97, 121)
(99, 68) (99, 95) (100, 52) (100, 111) (107, 80) (107, 83)
(109, 58) (109, 105) (110, 78) (110, 85) (111, 70) (111, 93)
(112, 12) (112, 151) (113, 61) (113, 102) (114, 57) (114, 106)
(115, 53) (115, 110) (117, 16) (117, 147) (118, 33) (118, 130)
(119, 48) (119, 115) (120, 22) (120, 141) (121, 6) (121, 157)
(122, 71) (122, 92) (125, 48) (125, 115) (126, 2) (126, 161)
(127, 47) (127, 116) (130, 39) (130, 124) (134, 14) (134, 149)
(135, 29) (135, 134) (136, 60) (136, 103) (140, 6) (140, 157)
(142, 70) (142, 93) (143, 72) (143, 91) (144, 28) (144, 135)
(148, 69) (148, 94) (149, 60) (149, 103) (151, 39) (151, 124)
(152, 17) (152, 146) (153, 28) (153, 135) (155, 40) (155, 123)
(156, 38) (156, 125) (158, 69) (158, 94) (159, 52) (159, 111)
(160, 42) (160, 121)

 First Random Point P  = (4, 45), 2P = (32, 110)
Second Random Point Q  = (4, 118), P+Q = (0, 0)
P += Q = (0, 0)
P += P = 2P = (32, 110)
```

Figure 6.5: Snapshot of the Demonstration of 32-bit ECC based Proposed Scheme.



```
Elliptic Curve Message Encryption Scheme
===============================================

G = (90, 127), order(G) is 31
Node A's Public Key Pa = 32*(90, 127) = (52, 36)
Node B's Public Key Pb = 131*(90, 127) = (62, 133)


Plain Text Message from Node A to Node B = (19, 72)

 Encrypted Message from Node A to Node B = (Pa,c1,c2) = ((52, 36), 26, 58)

  Node B's Decrypted Message from Node A = (19, 72)

Time elapsed: 335 ms
```

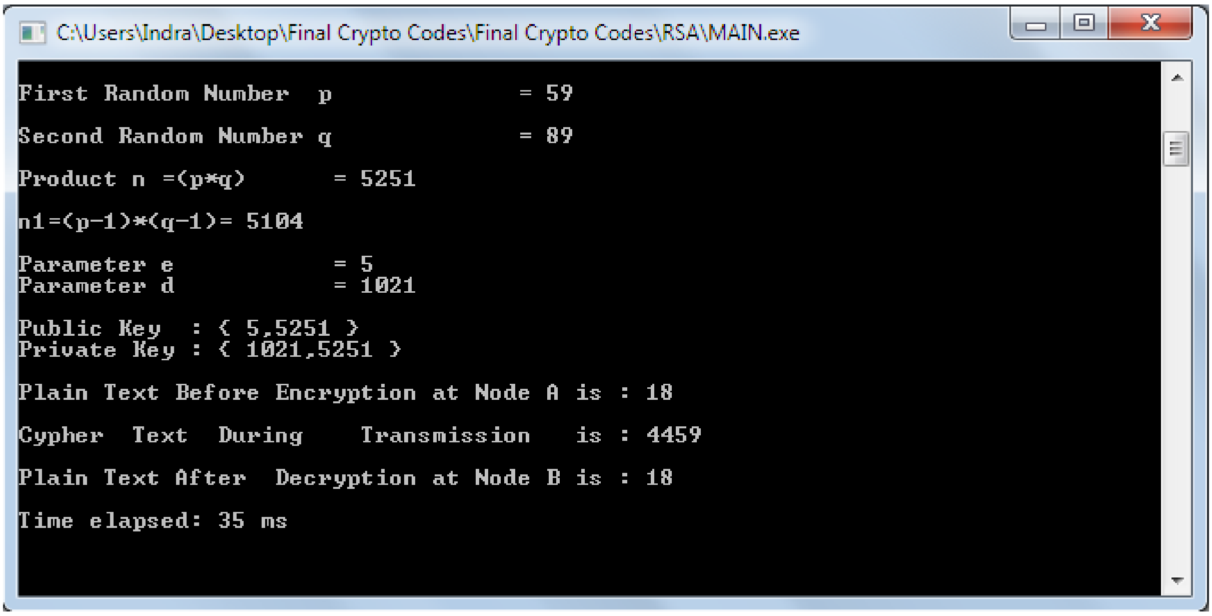
Figure 6.6: Snapshot of the Elliptic Curve Encryption Scheme from the Proposed Scheme.



C:\Users\Indra\Desktop\Final Crypto Codes\Final Crypto Codes\RSA\MAIN.exe

```
First Random Number  p              = 109

Second Random Number q              = 193

Product n =(p*q)        = 21037

n1=(p-1)*(q-1)= 20736

Parameter e             = 89
Parameter d             = 233

Public Key  : { 89,21037 }
Private Key : { 233,21037 }

Plain Text Before Encryption at Node A is : 18

Cypher  Text  During   Transmission   is : 13786

Plain Text After  Decryption at Node B is : 18

Time elapsed: 10 ms
```

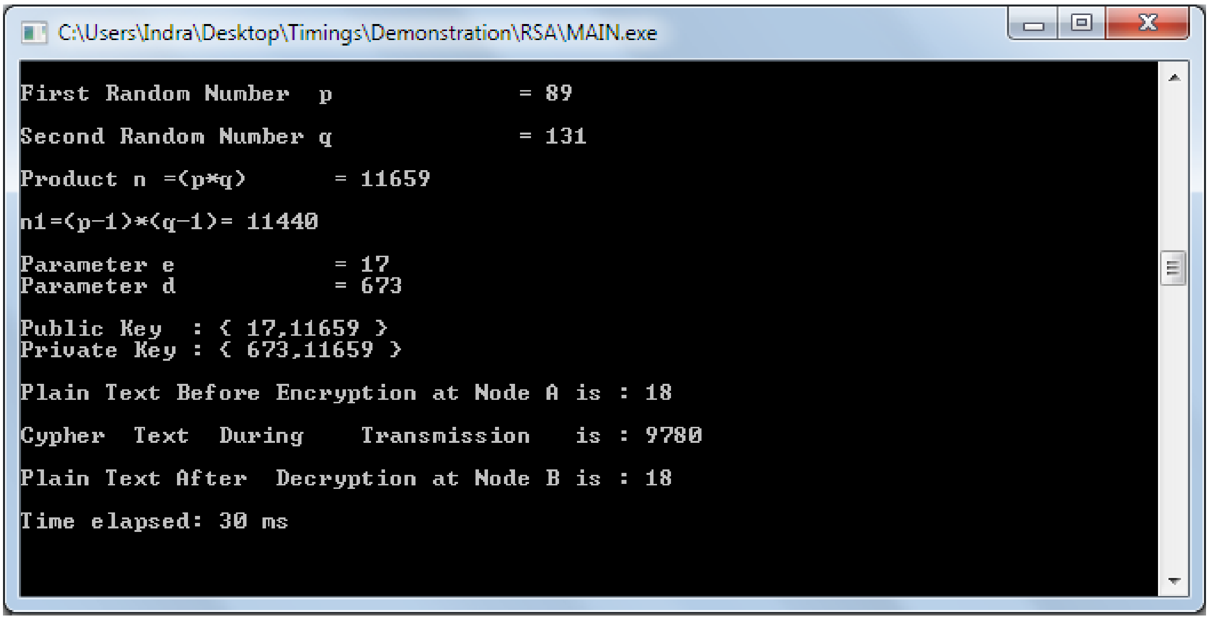
Figure 6.7: Snapshot of the Iteration No.1 for 32-bit RSA based Characteristic Encryption-Decryption Phase from the Proposed Scheme.



C:\Users\Indra\Desktop\Final Crypto Codes\Final Crypto Codes\RSA\MAIN.exe

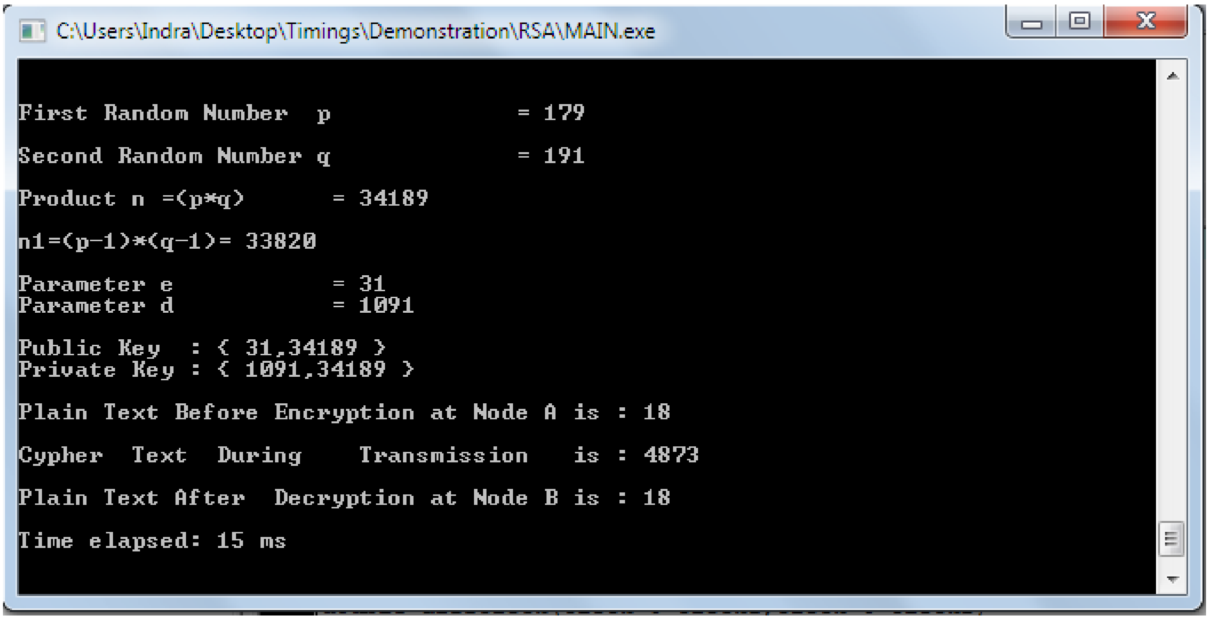
```
First Random Number  p              = 59

Second Random Number q              = 89

Product n =(p*q)        = 5251

n1=(p-1)*(q-1)= 5104

Parameter e             = 5
Parameter d             = 1021

Public Key  : { 5,5251 }
Private Key : { 1021,5251 }

Plain Text Before Encryption at Node A is : 18

Cypher  Text  During   Transmission   is : 4459

Plain Text After  Decryption at Node B is : 18

Time elapsed: 35 ms
```

Figure 6.8: Snapshot of the Iteration No.2 for 32-bit RSA based Characteristic Encryption-Decryption Phase from the Proposed Scheme.



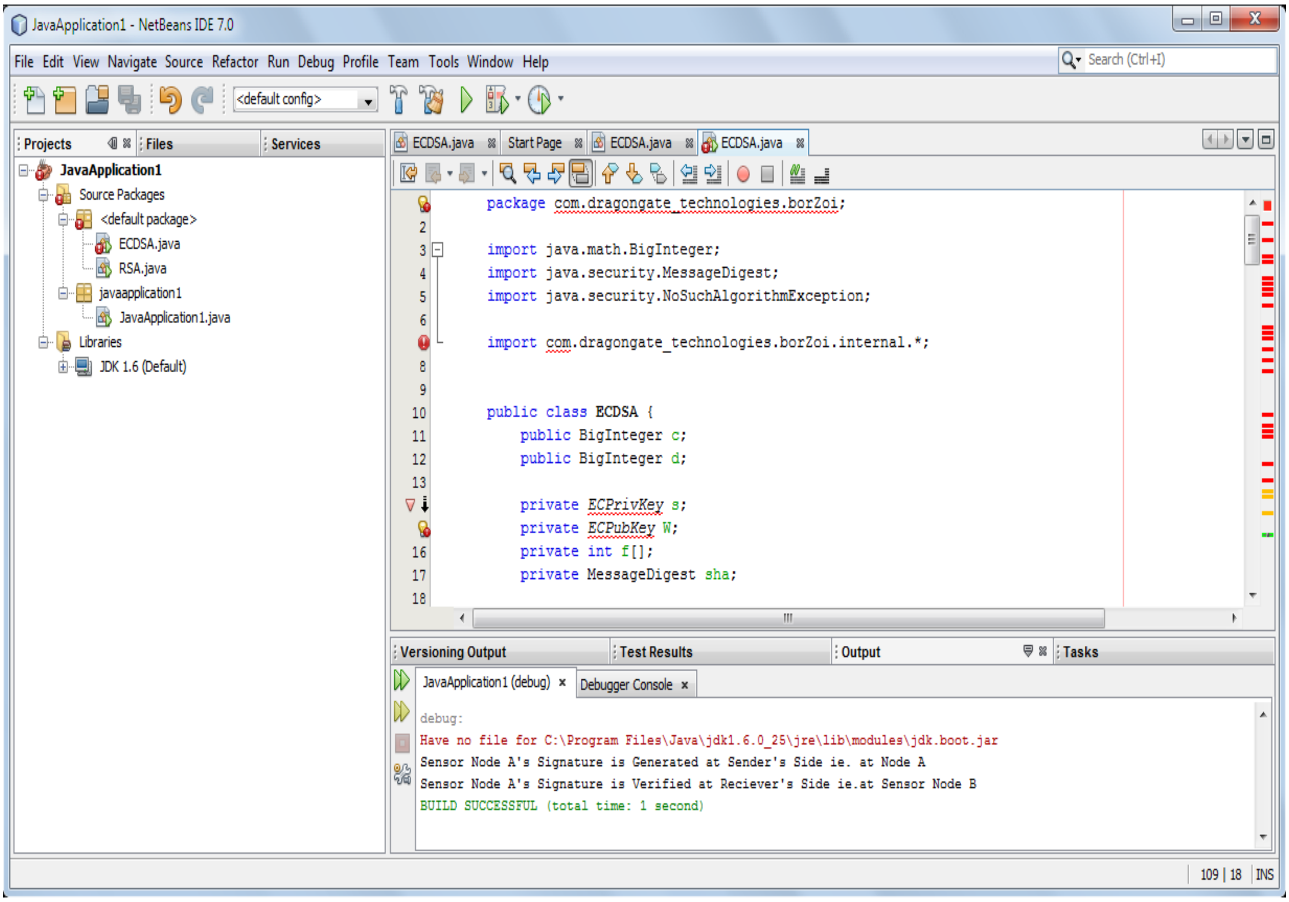
```
First Random Number  p              = 89
Second Random Number q              = 131
Product n =(p*q)       = 11659
n1=(p-1)*(q-1)= 11440
Parameter e            = 17
Parameter d            = 673
Public Key  : { 17,11659 }
Private Key : { 673,11659 }
Plain Text Before Encryption at Node A is : 18
Cypher  Text  During    Transmission   is : 9780
Plain Text After  Decryption at Node B is : 18
Time elapsed: 30 ms
```

Figure 6.9: Snapshot of the Iteration No.3 for 32-bit RSA based Characteristic Encryption-Decryption Phase from the Proposed Scheme.



```
First Random Number  p              = 179
Second Random Number q              = 191
Product n =(p*q)       = 34189
n1=(p-1)*(q-1)= 33820
Parameter e            = 31
Parameter d            = 1091
Public Key  : { 31,34189 }
Private Key : { 1091,34189 }
Plain Text Before Encryption at Node A is : 18
Cypher  Text  During    Transmission   is : 4873
Plain Text After  Decryption at Node B is : 18
Time elapsed: 15 ms
```

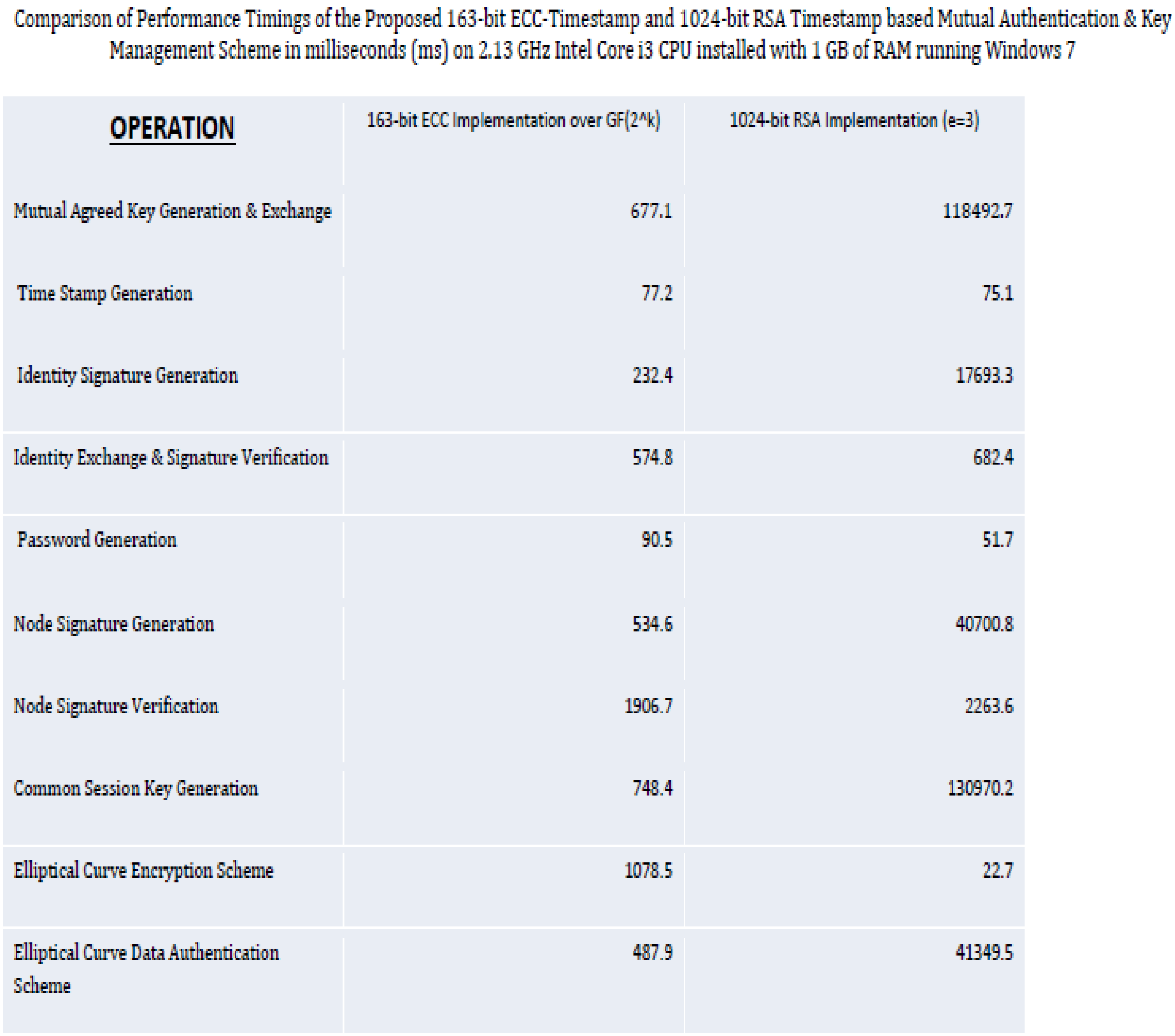
Figure 6.10: Snapshot of the Iteration No.4 for 32-bit RSA based Characteristic Encryption-Decryption Phase from the Proposed Scheme.



```
package com.dragongate_technologies.borZoi;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

import com.dragongate_technologies.borZoi.internal.*;


public class ECDSA {
    public BigInteger c;
    public BigInteger d;

    private ECPrivKey s;
    private ECPubKey W;
    private int f[];
    private MessageDigest sha;
```

```
debug:
Have no file for C:\Program Files\Java\jdk1.6.0_25\jre\lib\modules\jdk.boot.jar
Sensor Node A's Signature is Generated at Sender's Side ie. at Node A
Sensor Node A's Signature is Verified at Reciever's Side ie.at Sensor Node B
BUILD SUCCESSFUL (total time: 1 second)
```

Figure 6.11: Snapshot of the Characteristic Phase ECDSA from the Proposed Scheme.



Comparison of Performance Timings of the Proposed 163-bit ECC-Timestamp and 1024-bit RSA Timestamp based Mutual Authentication & Key Management Scheme in milliseconds (ms) on 2.13 GHz Intel Core i3 CPU installed with 1 GB of RAM running Windows 7

OPERATION	163-bit ECC Implementation over GF(2^k)	1024-bit RSA Implementation (e=3)
Mutual Agreed Key Generation & Exchange	677.1	118492.7
Time Stamp Generation	77.2	75.1
Identity Signature Generation	232.4	17693.3
Identity Exchange & Signature Verification	574.8	682.4
Password Generation	90.5	51.7
Node Signature Generation	534.6	40700.8
Node Signature Verification	1906.7	2263.6
Common Session Key Generation	748.4	130970.2
Elliptical Curve Encryption Scheme	1078.5	22.7
Elliptical Curve Data Authentication Scheme	487.9	41349.5

Table 6.1: Comparison of Performance Timings of 163-bit ECC-Timestamp and 1024-bit RSA Timestamp based Mutual Authentication & Key Mgt. Scheme.

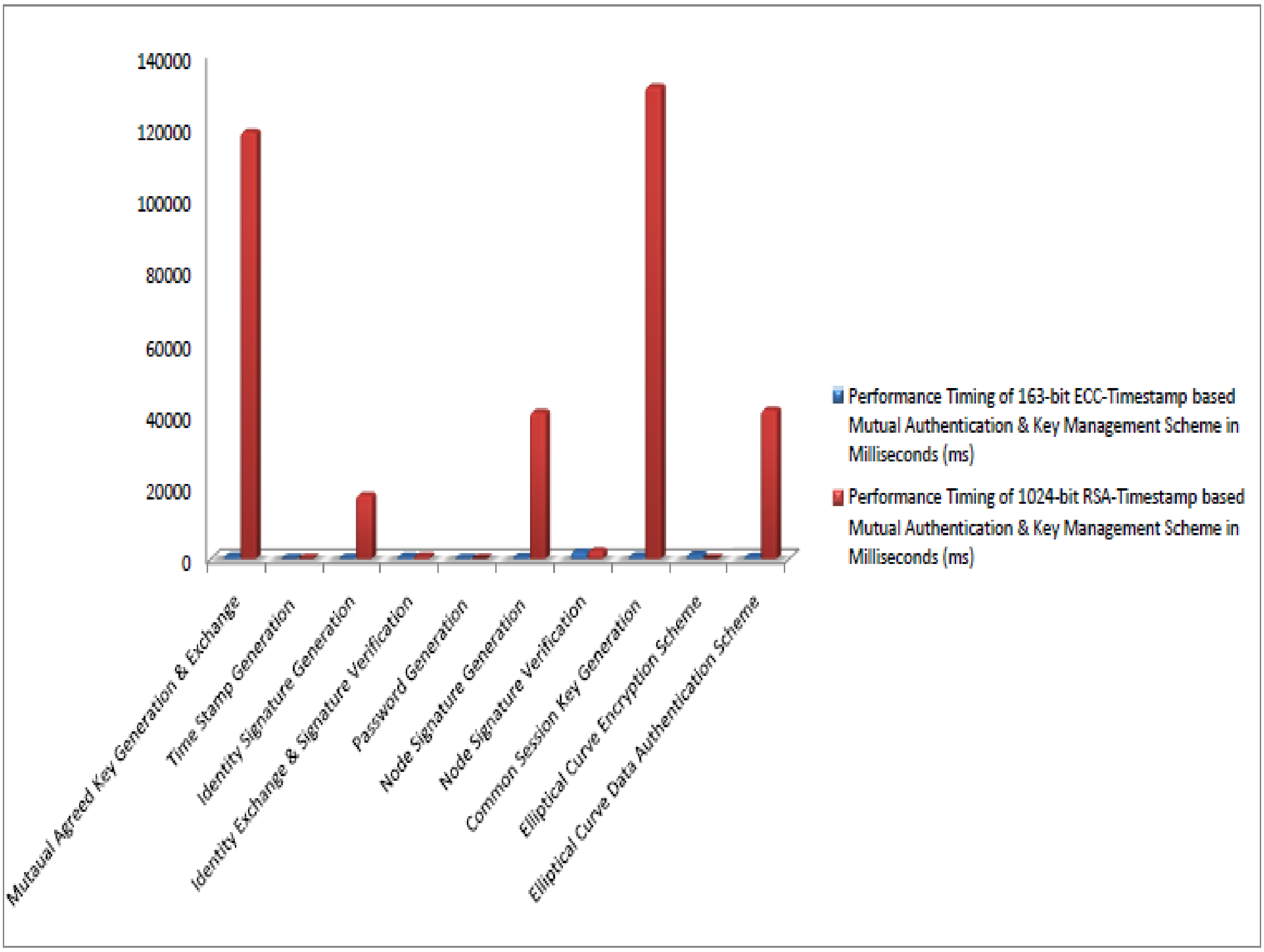


Figure 6.12: Graphical Comparison of Performance Timings of 163-bit ECC-Timestamp and 1024-bit RSA Timestamp based Mutual Authentication & Key Mgt. Scheme.

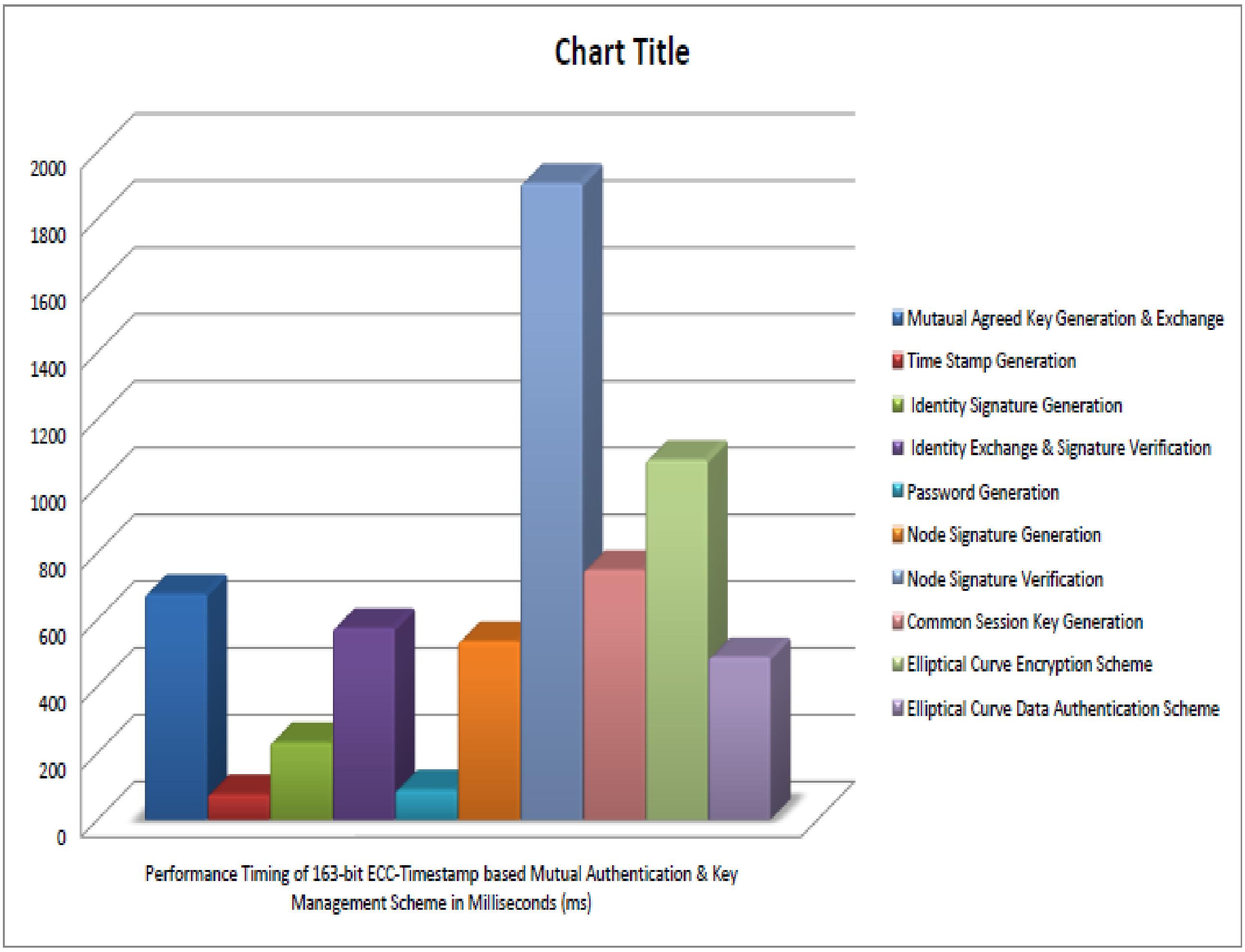


Figure 6.13: Performance Timings of 163-bit ECC-Timestamp based Mutual Authentication & Key Mgt. Scheme in Milliseconds (ms).

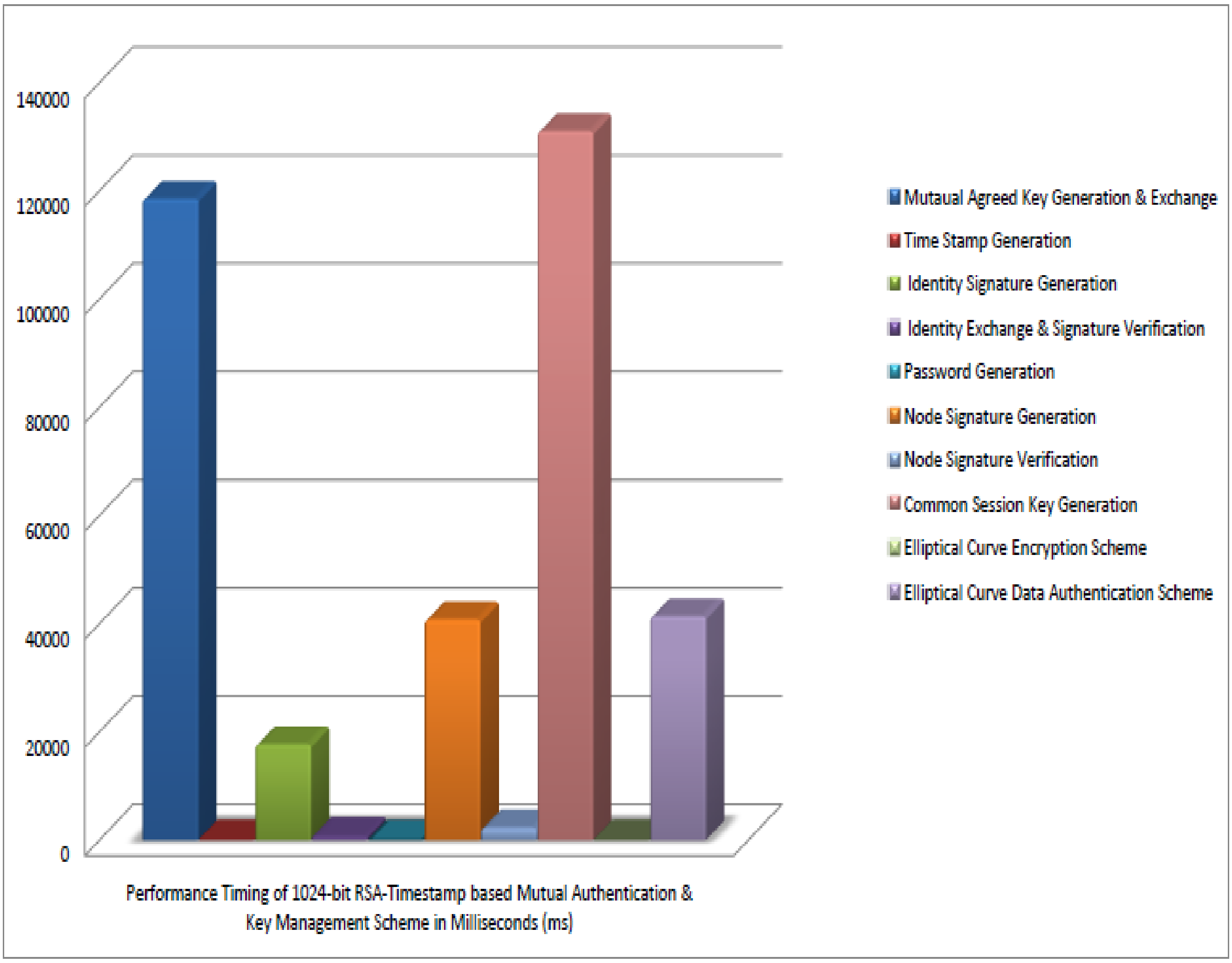


Figure 6.14: Performance Timing of 1024-bit RSA-Timestamp based Mutual Authentication & Key Mgt. Scheme in Milliseconds (ms).

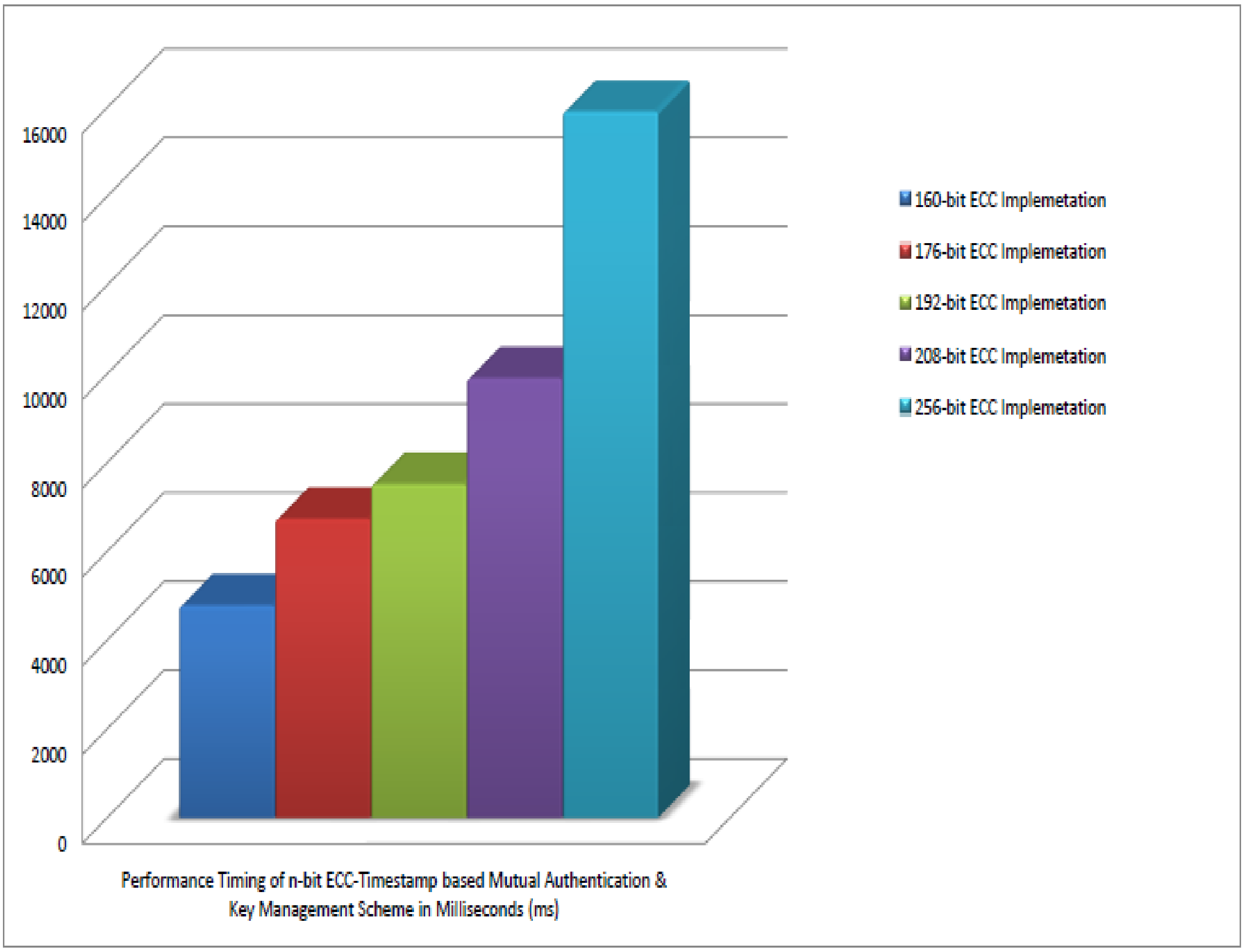


Figure 6.15: Comparison of Performance Timings of different n-bit ECC-Timestamp based Mutual Authentication & Key Management Schemes in Milliseconds (ms).

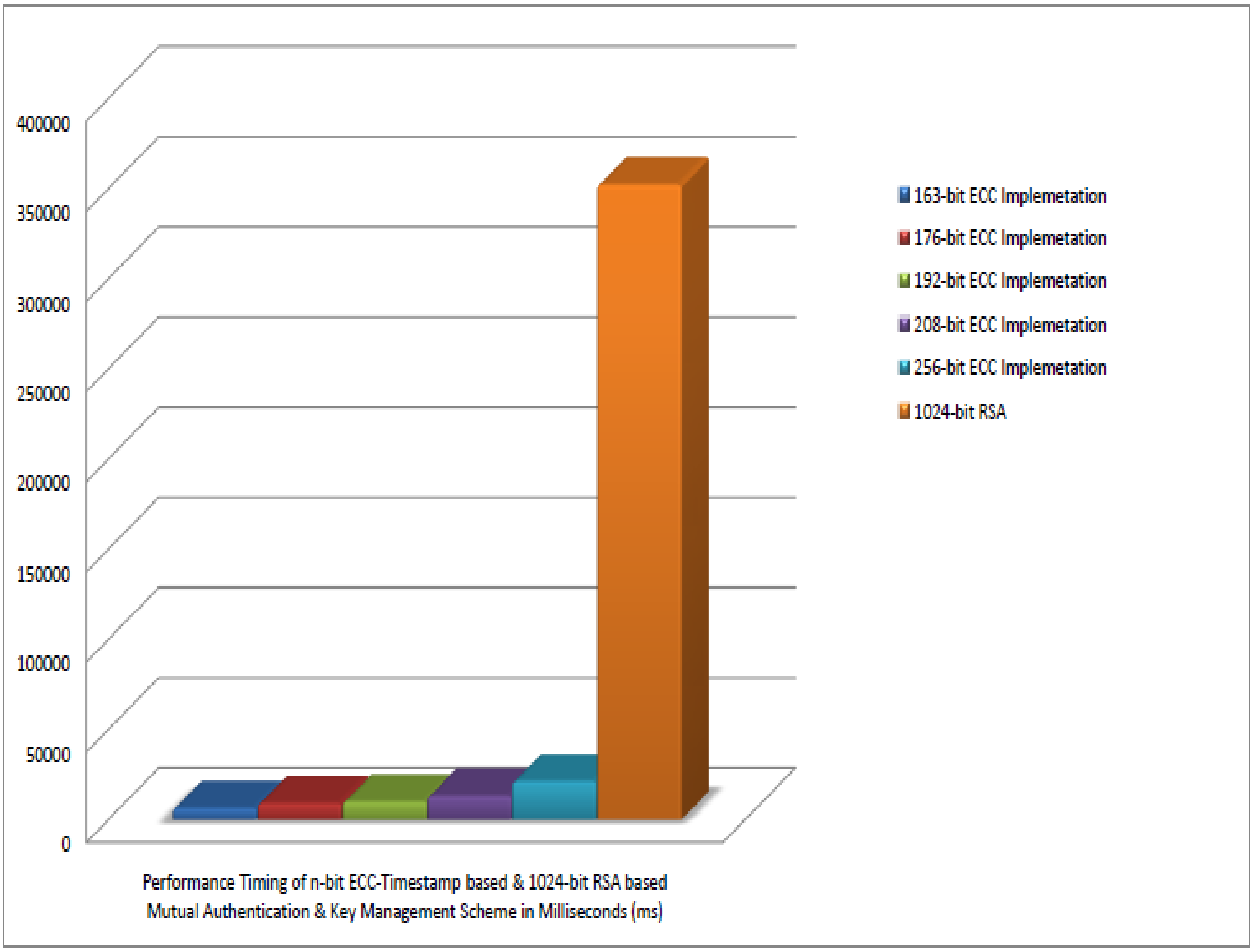


Figure 6.16: Mutual Comparison of Performance Timings of different n-bit ECC-Timestamp based & 1024-bit RSA based Mutual Authentication & Key Mgt. Schemes in Milliseconds (ms).

CHAPTER 7: CONCLUSION

Although there have been many efficient trusted server scheme, self enforcing scheme, and key pre-distribution schemes in the literature for general key agreement in the domain of Wireless Sensor Networks but the problem and limitation in terms of memory constraints and higher processing overhead along with insecure node communication posses a great threat to the reliability and conformity of Wireless Sensor Nodes in distributed environment. So in order to overcome these limitations and constraints we have proposed an ECC-Timestamp based Mutual Authentication and Key Management Scheme for WSNs which is basically an asymmetric key cryptographic technique with the feature of secure signature generation process and which in turn secures and protects the integrity of exchanged mutual agreed session key in different phases.

The proposed protocol or rather scheme is based on the Elliptic Curve Arithmetic employed in the corresponding cryptographic scheme and inherits the security and implementation characteristic of the same, which in turn seem to offer the highest cryptographic strength per bit among all existing public key cryptosystems. With a 160-bit modulus, an Elliptic Curve System seems to offer the same level of cryptographic security as standard DSA or RSA based cryptographic System with 1024-bit modulus and 224-bit ECC System is equivalent to 2048-bit RSA based Cryptographic System. The corresponding smaller key sizes result in smaller and efficient sensor node parameters along with smaller public key signatures and passwords, and also result in faster execution of the implemented scheme that too with low power requirements and the lesser processor memory requirements resulting in saving of the corresponding session bandwidth. Further the proposed scheme had also been able to counter and resist the very popular attacks on the wireless sensor networks to a great extent and had also been able to overcome the drawbacks of the loosely synchronised local clocks of the corresponding participants' viz. Node A and Node B respectively in an established session. In addition to this, the proposed scheme has also the provision of secure password generation for an efficient mutual authentication and verification of the respective participants.

CHAPTER 8: FUTURE WORK

Although Elliptic Curve Cryptography (ECC) is having good potential for wireless sensor network security due to its smaller key size and its high strength of security in the proposed scheme but still there is a room to reduce key calculation time to meet the potential applications in particular for wireless sensor networks. Scalar multiplication is the operation in elliptical curve cryptography which takes 80 % of key calculation time on wireless sensor network motes. There had been research proposals in the literature with algorithm based on 1’s complement subtraction to represent scalar in scalar multiplication which offer less Hamming weight and will remarkably improve the computational efficiency of scalar multiplication.

The positive integer in point multiplication employed in the scheme may be recoded with one’ complement subtraction to reduce the computational cost involved in this heavy mathematical operation for wireless sensor network platforms. 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 inversion operation to get the NAF of binary number, the one’s complement subtraction can provide a very simple way of recoding integer.

Apart from this, modular multiplication can be used in place of scalar point multiplication in the proposed scheme based on ECC thereby reducing the processing and memory overheads. This can be achieved by optimizing multiplication strategy in Elliptic Curve Cryptography. Further, efficient and appropriate modifications are required in the proposed scheme based on the standard Elliptic Curve Arithmetic in order to counter the Side Channel Attack.

Nevertheless the same proposed mutual authentication and key management scheme for a particular session in wireless sensor networks can be extended efficiently for a multi-session scenario in domain of wireless sensor networks or in the wired or wireless ad-hoc networks.

CHAPTER 9: REFERENCES AND BIBLIOGRAPHY

1. **Ian F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci**, *"Wireless Sensor Networks: A Survey."* Computer Networks Elsevier Journal, Vol. 38, No. 4, pp. 393-422, March 2002.

2. **Römer, Kay; Friedemann Mattern** (December 2004), *"The Design Space of Wireless Sensor Networks." IEEE Wireless Communications*11(6):54–61, doi:10.1109/MWC.2004.1368897

3. **Thomas Haenselmann** (2006-04-05), "*Sensornetworks*." GFDL Wireless Sensor Network textbook, http://pi4.informatik.uni-mannheim.de/~haensel/sn_book, retrieved 2006-08-29.

4. **Tiwari, Ankit** et. al, *"Energy-efficient wireless sensor network design and implementation for condition-based maintenance."* ACM Transactions on Sensor Networks (TOSN).

5. **Hadim, Salem; Nader Mohamed** (2006), *"Middleware Challenges and Approaches for Wireless Sensor Networks", IEEE Distributed Systems Online* 7 (3):1,http://doi.ieeecomputersociety.org/10.1109/MDSO.2006.19

6. **Zhang Li-Ping; Wang Yi; Li Gui-Ling;** *"A Novel Group Key Agreement Protocol for Wireless Sensor Networks";* Wireless Communications & Signal Processing, 2009. WCSP 2009. International Conference; Publication Year: 2009 , Page(s): 1 – 4.

7. **Pritam Gajkumar Shah, Xu Huang, Dharmendra Sharma**,*" Analytical study of implementation issues of Elliptical Curve Cryptography for Wireless Sensor networks";* 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Wo

8. **Pradnya Gajbhiye, Anjali Mahajan** ;" *A Survey of Architecture and Node deployment in Wireless Sensor Network "*; Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference; Publication Year: 2008 , Page(s): 426 – 430.

9. **Hasan Tahir, Syed Asim Ali Shah***; "Wireless Sensor Networks*- *A Security Perspective."*2008. Proceedings of the 12th IEEE International Multitopiv Conference, December 23-24, 2008.

10. **Ismail, M.; Sanavullah, M.Y***.;" Security topology in wireless sensor networks with routing optimization",* Wireless Communication and Sensor Networks, 2008. WCSN 2008. Fourth International Conference on Digital Object Identifier: 10.1109/WCSN.2008.4772673 Publication Year: 2008 , Page(s): 7 – 15.

11. **Pritam Gajkumar Shah, Xu Huang, Dharmendra Sharma** ; *"Algorithm based on one's complement for fast scalar multiplication in ECC for Wireless Sensor Network";* 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops.

12. **Romer, K.; Mattern, F.***;" The design space of wireless sensor networks".* Wireless Communications, IEEE Volume: 11 ,Issue; Publication Year: 2004 , Page(s):54 – 61.

13. **Zhan Liu and Mi Lu;** *"Time Stamp Counter Mechanism and Application on Sensor Networks."* Systems, Applications and Technology Conference, 2006. LISAT 2006. IEEE Long Island; Publication Year: 2006 , Page(s): 1 - 6

14. **Farzad Kiani and Gokhan Dalkilic,** Computer Engineering Department, Dokuz Eylul University, Izmir, Turkey *"Password Renewal Enhancement for Dynamic Authentication in Wireless Sensor Networks."* 2010 Second International Conference

on Computational Intelligence, Communication Systems and Networks. ICCI 2010; Publication Year 2010.

15. **Chatterjee, K.; Gupta, D***;"Secure Access of Smart Cards using Elliptic Curve Cryptosystems."*
    Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference.
    Publication Year: 2009 , Page(s): 1 – 4

16. **H. Wang, B. Sheng, and Q. Li,** *"Elliptic curve cryptography-based access control in sensor networks,"* Int. J. Security and Networks,, vol. 1, pp. 127-137, 2006.

17. **W. Du, J. Deng, Y. S. Han, and P. K. Varshney**. *"A pairwise key pre-distribution scheme for wireless sensor networks,"* in 10th ACM Conference on Computer and Communications Security (CCS), pages 42–51, Washington, DC, USA, 2003.

18. **N.Koblitz**, *"Elliptic Curve Cryptosystems,"* Mathematics of Computation, vol. 48, pp. 203-209, 1987.

19. **V. S. Miller**, *"Use of Elliptic Curves in Cryptography,"* in Advances in Cryptology - CRYPTO '85: Proceedings. vol. 218:Springer-Verlag, 1986, pp. 417-426.

20. **J. D.Tygar**, et al, "SPINS: SecurityProtocols for Sensor Networks", Wireless Networks, v8, n5, Sept. 2002, pp. 521 -534.

21. **L.EschenauerandVirgilD.Gligor**, *"A key-management scheme for distributed sensor networks",* Prodeedings of the ACM Conference on Computer and Communication Security, 2002,p41 -47.

22. **Seyit A. Camtepe**, et al, *"Key distribution mechanisms for wireless sensor networks: a survey"*, TR-05-07, Dept. of Computer Science, Rensselaer Polytechnic Institute, March23, 2005.

23. **S. A. Camtepe and B. Yener**., "*Combinatorial design of key distribution mechanisms for wireless sensor networks,"* in Proceedings of 9th European Symposium On Research in Computer Security (ESORICS'04), 2004.

24. **R. Syed and Sungyong Lee**, *"A Survey on Key Management Strategies for Different Applications of WSNs."* Journal of Computing Science and Engineering. Volume 4, No.1, Publication Year March 2010. Pages 23-51

25. **Wang, H and Q.Li,** *"Efficient Implementation of public key cryptosystem on mote sensors."* In International Conference on Information and Communication Society, LNCS 4307. Publication Year 2006.

APPENDIX A: SOURCE CODE

SOURCE CODE FOR 163-bit ECDSA:

```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.SecureRandom;
import java.security.Signature;
import java.security.spec.ECGenParameterSpec;

 public class ECDSA {
       public BigInteger c;
       public BigInteger d;

       private ECPrivKey s;
       private ECPubKey W;
       private int f[];
       private MessageDigest sha;

       private BigInteger[] ECSP_DSA() {
           BigInteger sig[] = { BigInteger.valueOf(0), BigInteger.valueOf(0)};
           ECPrivKey u;
           ECPubKey V;
           while ((sig[0].compareTo(BigInteger.valueOf(0)) == 0)
               || (sig[1].compareTo(BigInteger.valueOf(0)) == 0)) {
               u = new ECPrivKey(s.dp);
               V = new ECPubKey(u);
```

```
    sig[0] = Utils.OS2IP(Utils.FE2OSP(V.W.x)).mod(s.dp.r);
    BigInteger uinv = u.s.modInverse(s.dp.r);
    BigInteger temp =
      Utils.OS2IP(f).add(s.s.multiply(sig[0]).mod(s.dp.r)).mod(s.dp.r);
    sig[1] = (uinv.multiply(temp)).mod(s.dp.r);
  }

  return sig;
}

private boolean ECVP_DSA() {
  if (!(((BigInteger.valueOf(1).compareTo(c) = 0)
    & (c.compareTo(W.dp.r)  0))
    & ((BigInteger.valueOf(1).compareTo(d) = 0)
      & (d.compareTo(W.dp.r)  0))))
    return false;

  BigInteger h = d.modInverse(W.dp.r);
  BigInteger h1 = Utils.OS2IP(f).multiply(h).mod(W.dp.r);
  BigInteger h2 = c.multiply(h).mod(W.dp.r);
   ECPoint P = W.dp.E.add(W.dp.E.mul(h1, W.dp.G), W.dp.E.mul(h2, W.W));
   if (P.isZero())
    return false;
   BigInteger i = Utils.OS2IP(Utils.FE2OSP(P.x)).mod(W.dp.r);
  if (c.compareTo(i) == 0)
    return true;
  else
    return false;
}

public ECDSA() {
}

public ECDSA(BigInteger c, BigInteger d) {
   this.c = c;
   this.d = d;
}

public void initSignature(ECPrivKey s) throws NoSuchAlgorithmException {
   sha = MessageDigest.getInstance("SHA");
   this.s = (ECPrivKey) s.clone();
}

public void initVerify(ECPubKey W) throws NoSuchAlgorithmException {
   sha = MessageDigest.getInstance("SHA");
   this.W = (ECPubKey) W.clone();
}

public void update(byte[] data) {
   sha.update(data);
}

public void sign() {
   f = Utils.revIntArray(Utils.toIntArray(sha.digest()));
   BigInteger[] sig = ECSP_DSA();
   c = sig[0];
   d = sig[1];
}

public boolean verify() {
   f = Utils.revIntArray(Utils.toIntArray(sha.digest()));
   return ECVP_DSA();
```

```
        }


        public String toString() {
           String str = new String("c:").concat(c.toString(16)).concat("\n");
           str = str.concat("d:").concat(d.toString(16)).concat("\n");


           return str;
        }


        protected Object clone() {
           return new ECDSA();
        }


public static void main(
        String[]    args)
        throws Exception
    {
        KeyPairGenerator keyGen = KeyPairGenerator.getInstance("ECDSA", "BC");
        ECGenParameterSpec ecSpec = new ECGenParameterSpec("prime192v1");

        keyGen.initialize(ecSpec, new SecureRandom());

        KeyPair            keyPair = keyGen.generateKeyPair();
        Signature          signature = Signature.getInstance("ECDSA", "BC");
// generate a signature
        signature.initSign(keyPair.getPrivate(), Utils.createFixedRandom());
System.out.println ("Sensor Node A's Signature is Generated at Sender's Side ie. at Node
A");

        byte[] message = new byte[] { (byte)'a', (byte)'b', (byte)'c' };

        signature.update(message);

        byte[] sigBytes = signature.sign();

                // verify a signature
                signature.initVerify(keyPair.getPublic());
                signature.update(message);
        if (signature.verify(sigBytes))
        {
                        System.out.println("Sensor Node A's Signature is Verified at Reciever's Side
             ie.at Sensor Node B");
        }
        else
        {
           System.out.println("Sensor Node A's Signature Verification Failed at Reciever's Side
ie.at Sensor Node B.");
                                }
                }
}
```

SOURCE CODE FOR 32-bit ECDH & ECAES:

FiniteFieldElement.hpp

```
namespace Cryptography
{
      // helper functions
       namespace   detail
       {
          //From Knuth; Extended GCD gives g = a*u + b*v
          int EGCD(int a, int b, int& u, int &v)
          {
             u = 1;
             v = 0;
             int g = a;
             int u1 = 0;
             int v1 = 1;
             int g1 = b;
             while (g1 != 0)
             {
                int q = g/g1; // Integer divide
                int t1 = u - q*u1;
                int t2 = v - q*v1;
                int t3 = g - q*g1;
                u = u1; v = v1; g = g1;
                u1 = t1; v1 = t2; g1 = t3;
             }

             return g;
          }

          int InvMod(int x, int n) // Solve linear congruence equation x * z == 1 (mod n) for z
          {
             //n = Abs(n);
             x = x % n; // % is the remainder function, 0 <= x % n < |n|
             int u,v,g,z;
             g = EGCD(x, n, u,v);
             if (g != 1)
             {
                // x and n have to be relative prime for there to exist an x^-1 mod n
                z = 0;
             }
             else
             {
```

```
         z = u % n;
      }
      return z;
   }
}

template<int P>
class   FiniteFieldElement
{
   int     i_;

   void    assign(int i)
   {
      i_ = i;
      if ( i<0 )
      {
         // ensure (-i) mod p correct behaviour
         // the (i%P) term is to ensure that i is in the correct range before normalizing
         i_ = (i%P) + 2*P;
      }

      i_ %= P;
   }

   public:
      // ctor
      FiniteFieldElement()
       : i_(0)
      {}
      // ctor
      explicit FiniteFieldElement(int i)
      {
         assign(i);
      }
      // copy ctor
      FiniteFieldElement(const FiniteFieldElement<P>& rhs)
       : i_(rhs.i_)
      {
      }

      // access "raw" integer
      int i() const { return i_; }
      // negate
      FiniteFieldElement  operator-() const
      {
         return FiniteFieldElement(-i_);
```

```
            }
            // assign from integer
            FiniteFieldElement& operator=(int i)
            {
               assign(i);
               return *this;
            }
            // assign from field element
            FiniteFieldElement<P>& operator=(const FiniteFieldElement<P>& rhs)
            {
               i_ = rhs.i_;
               return *this;
            }
            // *=
            FiniteFieldElement<P>& operator*=(const FiniteFieldElement<P>& rhs)
            {
               i_ = (i_*rhs.i_) % P;
               return *this;
            }
            // ==
            friend bool    operator==(const FiniteFieldElement<P>& lhs, const
FiniteFieldElement<P>& rhs)
            {
               return (lhs.i_ == rhs.i_);
            }
            // == int
            friend bool    operator==(const FiniteFieldElement<P>& lhs, int rhs)
            {
               return (lhs.i_ == rhs);
            }
            // !=
            friend bool    operator!=(const FiniteFieldElement<P>& lhs, int rhs)
            {
               return (lhs.i_ != rhs);
            }
            // a / b
            friend FiniteFieldElement<P> operator/(const FiniteFieldElement<P>& lhs, const
FiniteFieldElement<P>& rhs)
            {
               return FiniteFieldElement<P>( lhs.i_ * detail::InvMod(rhs.i_,P));
            }
            // a + b
            friend FiniteFieldElement<P> operator+(const FiniteFieldElement<P>& lhs, const
FiniteFieldElement<P>& rhs)
            {
               return FiniteFieldElement<P>( lhs.i_ + rhs.i_);
```

```
        }
        // a - b
        friend FiniteFieldElement<P> operator-(const FiniteFieldElement<P>& lhs, const
FiniteFieldElement<P>& rhs)
        {
           return FiniteFieldElement<P>( lhs.i_ - rhs.i_);
        }
        // a + int
        friend FiniteFieldElement<P> operator+(const FiniteFieldElement<P>& lhs, int i)
        {
           return FiniteFieldElement<P>( lhs.i_+i);
        }
        // int + a
        friend FiniteFieldElement<P> operator+(int i, const FiniteFieldElement<P>& rhs)
        {
           return FiniteFieldElement<P>( rhs.i_+i);
        }
        // int * a
        friend FiniteFieldElement<P> operator*(int n, const FiniteFieldElement<P>& rhs)
        {
           return FiniteFieldElement<P>( n*rhs.i_);
        }
        // a * b
        friend FiniteFieldElement<P> operator*(const FiniteFieldElement<P>& lhs, const
FiniteFieldElement<P>& rhs)
        {
           return FiniteFieldElement<P>( lhs.i_ * rhs.i_);
        }
        // ostream handler
        template<int T>
        friend  ostream&    operator<<(ostream& os, const FiniteFieldElement<T>& g)
        {
           return os << g.i_;
        }
    };
}
```

indra.cpp

```
#include <cstdlib>
#include <iostream>
#include <vector>

using namespace std;

#include <math.h>
#include <ctime>
#include <conio.h>
#include "FiniteFieldElement.hpp"

double diffclock(clock_t clock1,clock_t clock2)
{
        double diffticks=clock1-clock2;
        double diffms=(diffticks*1000)/CLOCKS_PER_SEC;
        return diffms;
}

namespace Cryptography
{
    /*
      Elliptic Curve over a finite field of order P:
      y^2 mod P = x^3 + ax + b mod P

      Template parameter P is the order of the finite field Fp over which this curve is
defined
    */
    template<int P>
    class   EllipticCurve
    {
      public:
        // this curve is defined over the finite field (Galois field) Fp, this is the
        // typedef of elements in it
        typedef FiniteFieldElement<P> ffe_t;
        class   Point
        {
          friend  class   EllipticCurve<P>;
          typedef FiniteFieldElement<P> ffe_t;
          ffe_t  x_;
          ffe_t  y_;
          EllipticCurve    *ec_;

          void   addDouble(int m, Point& acc)
          {
```

```
   if ( m > 0 )
   {
      Point r = acc;
      for ( int n=0; n < m; ++n )
      {
         r += r;
      }
      acc = r;
   }
}

Point scalarMultiply(int k, const Point& a)
{
   Point acc = a;
   Point res = Point(0,0,*ec_);
   int i = 0, j = 0;
   int b = k;

   while( b )
   {
      if ( b & 1 )
      {

         addDouble(i-j,acc);
         res += acc;
         j = i;
      }
      b >>= 1;
      ++i;
   }
   return res;
}
// adding two points on the curve
void    add(ffe_t x1, ffe_t y1, ffe_t x2, ffe_t y2, ffe_t & xR, ffe_t & yR) const
{

   if ( x1 == 0 && y1 == 0 )
   {
      xR = x2;
      yR = y2;
      return;
   }
   if ( x2 == 0 && y2 == 0 )
   {
      xR = x1;
      yR = y1;
      return;
   }
   if ( y1 == -y2 )
   {
      xR = yR = 0;
      return;
   }


   ffe_t s;
   if ( x1 == x2 && y1 == y2 )
   {
      //2P
      s = (3*(x1.i()*x1.i()) + ec_->a()) / (2*y1);
      xR = ((s*s) - 2*x1);
   }
   else
   {
      //P+Q
      s = (y1 - y2) / (x1 - x2);
      xR = ((s*s) - x1 - x2);
   }

   if ( s != 0 )
   {
      yR = (-y1 + s*(x1 - xR));
   }
   else
   {
      xR = yR = 0;
   }
}

Point(int x, int y)
: x_(x),
  y_(y),
  ec_(0)
{}

Point(int x, int y, EllipticCurve<P> & EllipticCurve)
 : x_(x),
   y_(y),
   ec_(&EllipticCurve)
{}

Point(const ffe_t& x, const ffe_t& y, EllipticCurve<P> & EllipticCurve)
```

```
    : x_(x),
      y_(y),
      ec_(&EllipticCurve)
   {}

public:
   static  Point   ONE;

   // copy ctor
   Point(const Point& rhs)
   {
      x_ = rhs.x_;
      y_ = rhs.y_;
      ec_ = rhs.ec_;
   }
   // assignment
   Point& operator=(const Point& rhs)
   {
      x_ = rhs.x_;
      y_ = rhs.y_;
      ec_ = rhs.ec_;
      return *this;
   }
   // access x component as element of Fp
   ffe_t x() const { return x_; }
   // access y component as element of Fp
   ffe_t y() const { return y_; }

   unsigned int     Order(unsigned int maxPeriod = ~0) const
   {
      Point r = *this;
      unsigned int n = 0;
      while( r.x_ != 0 && r.y_ != 0 )
      {
         ++n;
         r += *this;
         if ( n > maxPeriod ) break;
      }
      return n;
   }
   // negate
   Point   operator-()
   {
      return Point(x_,-y_);
   }
   // ==
   friend bool    operator==(const Point& lhs, const Point& rhs)
   {
      return (lhs.ec_ == rhs.ec_) && (lhs.x_ == rhs.x_) && (lhs.y_ == rhs.y_);
   }
   // !=
   friend bool    operator!=(const Point& lhs, const Point& rhs)
   {
      return (lhs.ec_ != rhs.ec_) || (lhs.x_ != rhs.x_) || (lhs.y_ != rhs.y_);
   }
   // a + b
   friend Point operator+(const Point& lhs, const Point& rhs)
   {
      ffe_t xR, yR;
      lhs.add(lhs.x_,lhs.y_,rhs.x_,rhs.y_,xR,yR);
      return Point(xR,yR,*lhs.ec_);
   }
   // a * int
   friend  Point operator*(int k, const Point& rhs)
   {
      return Point(rhs).operator*=(k);
   }
   // +=
   Point& operator+=(const Point& rhs)
   {
      add(x_,y_,rhs.x_,rhs.y_,x_,y_);
      return *this;
   }
   // a *= int
   Point& operator*=(int k)
   {
      return (*this = scalarMultiply(k,*this));
   }
   // ostream handler: print this point
   friend ostream& operator <<(ostream& os, const Point& p)
   {
      return (os << "(" << p.x_ << ", " << p.y_ << ")");
   }
};

// Elliptic Curve Implementation

typedef EllipticCurve<P> this_t;
typedef class EllipticCurve<P>::Point point_t;

// Initialize EC as y^2 = x^3 + ax + b
EllipticCurve(int a, int b)
```

```
: a_(a),
 b_(b),
 m_table_(),
 table_filled_(false)
{
}

void    CalculatePoints()
{
  int x_val[P];
  int y_val[P];
  for ( int n = 0; n < P; ++n )
  {
    int nsq = n*n;
    x_val[n] = ((n*nsq) + a_.i() * n + b_.i()) % P;
    y_val[n] = nsq % P;
  }

  for ( int n = 0; n < P; ++n )
  {
    for ( int m = 0; m < P; ++m )
    {
      if ( x_val[n] == y_val[m] )
      {
        m_table_.push_back(Point(n,m,*this));
      }
    }
  }

  table_filled_ = true;
}

Point   operator[](int n)
{
  if ( !table_filled_ )
  {
    CalculatePoints();
  }

  return m_table_[n];
}

size_t  Size() const { return m_table_.size(); }

int     Degree() const { return P; }
```

```
      FiniteFieldElement<P>  a() const { return a_; }

      FiniteFieldElement<P>  b() const { return b_; }


      template<int T>
      friend ostream& operator <<(ostream& os, const EllipticCurve<T>& EllipticCurve);

      ostream&    PrintTable(ostream &os, int columns=4);

      private:
         typedef std::vector<Point>  m_table_t;

         m_table_t                 m_table_;   // table of points
         FiniteFieldElement<P>      a_;        // paramter a of the EC equation
         FiniteFieldElement<P>      b_;        // parameter b of the EC equation
         bool    table_filled_;                // true if the table has been calculated
};

template<int T>
   typename EllipticCurve<T>::Point EllipticCurve<T>::Point::ONE(0,0);

template<int T>
ostream& operator <<(ostream& os, const EllipticCurve<T>& EllipticCurve)
{
   os << "y^2 mod " << T << " = (x^3" << showpos;
   if ( EllipticCurve.a_ != 0 )
   {
      os << EllipticCurve.a_ << "x";
   }

   if ( EllipticCurve.b_.i() != 0 )
   {
      os << EllipticCurve.b_;
   }

   os << noshowpos << ") mod " << T;
   return os;
}

template<int P>
ostream&    EllipticCurve<P>::PrintTable(ostream &os, int columns)
{
   if ( table_filled_ )
   {
      int col = 0;
```

```
            typename EllipticCurve<P>::m_table_t::iterator iter = m_table_.begin();
            for ( ; iter!=m_table_.end(); ++iter )
            {
               os << "(" << (*iter).x_.i() << ", " << (*iter).y_.i() << ") ";
               if ( ++col > columns )
               {
                  os << "\n";
                  col = 0;
               }
            }
         }
         else
         {
            os << "EllipticCurve, F_" << P;
         }
         return os;
      }
}

namespace   utils
{
   float   frand()
   {
      static float norm = 1.0f / (float)RAND_MAX;
      return (float)rand()*norm;
   }

   int irand(int min, int max)
   {
      return min+(int)(frand()*(float)(max-min));
   }
}

using namespace Cryptography;
using namespace utils;

int main(int argc, char *argv[])
{
   typedef EllipticCurve<163> ec_t;
   ec_t   myEllipticCurve(1,1);


   cout << "An ECC-Timestamp based Mutual Authentication & Key Management Scheme for
WSNs\n\n";
   cout << "A Session Demonstration for 32-bit ECC based Proposed Scheme\n\n\n\n";
```

```
    cout << "The elliptic curve: " << myEllipticCurve << "\n";

    clock_t begin=clock();

    myEllipticCurve.CalculatePoints();


    cout << "\nPoints on the curve (i.e. the group elements):\n";
    myEllipticCurve.PrintTable(cout,5);
    cout << "\n\n";

    ec_t::Point P = myEllipticCurve[2];
    cout << " First Random Point P  = " << P << ", 2P = " << (P+P) << "\n";
    ec_t::Point Q = myEllipticCurve[3];
    cout << "Second Random Point Q  = " << Q << ", P+Q = " << (P+Q) << "\n";
    ec_t::Point R = P;
    R += Q;
    cout << "P += Q = " << R << "\n";
    R = P;
    R += R;
    cout << "P += P = 2P = " << R << "\n";

    cout << "\nElliptic Curve Message Encryption
Scheme\n====================================================\n\n";

    // Menes-Vanstone EC message encryption scheme


    ec_t::Point G = myEllipticCurve[0];
    while( (G.y() == 0 || G.x() == 0) || (G.Order()<2) )
    {
       int n = (int)(frand()*myEllipticCurve.Size());
       G = myEllipticCurve[n];
    }

    cout << "G = " << G << ", order(G) is " << G.Order() << "\n";

    // Alice
    int a = irand(1,myEllipticCurve.Degree()-1);
    ec_t::Point Pa = a*G;  // public key
    cout << "Node A's Public Key Pa = " << a << "*" << G << " = " << Pa << endl;

    // Bob
    int b = irand(1,myEllipticCurve.Degree()-1);;
    ec_t::Point Pb = b*G;  // public key
    cout << "Node B's Public Key Pb = " << b << "*" << G << " = " << Pb << endl;
```

```
   int j = irand(1,myEllipticCurve.Degree()-1);;
   ec_t::Point Pj = j*G;
  // cout << "Jane's public key Pj = " << j << "*" << G << " = " << Pj << endl;

   cout << "\n\n";


   int m1 = 19;
   int m2 = 72;

   cout << "Plain Text Message from Node A to Node B = (" << m1 << ", " << m2 << ")\n\n";

   // encrypt using Bob`s key
   ec_t::Point Pk = a*Pb;
   ec_t::ffe_t c1( m1*Pk.x() );
   ec_t::ffe_t c2( m2*Pk.y() );

   // encrypted message is: Pa,c1,c2
   cout << " Encrypted Message from Node A to Node B = {Pa,c1,c2} = {" << Pa << ", " << c1
<< ", " << c2 << "}\n\n";

   Pk = b*Pa;
   ec_t::ffe_t m1d = c1/Pk.x();
   ec_t::ffe_t m2d = c2/Pk.y();

   cout << "  Node B's Decrypted Message from Node A = (" << m1d << ", " << m2d << ")" <<
endl;

   Pk = j*Pa;
   m1d = c1/Pk.x();
   m2d = c2/Pk.y();

   //cout << "\nJane's decrypted message from Alice = (" << m1d << ", " << m2d << ")" <<
endl;

   cout << endl;
   clock_t end=clock();

            cout << "Time elapsed: " << double(diffclock(end,begin)) << " ms"<< endl;
            getch();
   system("PAUSE");
   return EXIT_SUCCESS;
}
```