Energy Analysis of WSN using RSA and ECC Encryption method

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Sharanjit Kaur
(Roll No. 801032022)

Under the supervision of:
Dr. Maninder Singh
Associate Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

May 2012

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, **"Energy Analysis of WSN using RSA and ECC Encryption method"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Maninder Singh and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Sharanjit Kaur

(Sharanjit Kaur)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Maninder Singh)
Associate Professor
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

(Dr. Maninder Singh)
Associate Professor and Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

ACKNOWLEDGMENT

First of all I am thankful to God for blessings and showing me the right decision. With his mercy, it has been possible for me to reach so far.

I would like to express sincerest thanks to my thesis supervisor Dr. Maninder Singh, for his inspiration, guidance, stimulating suggestions, immense help and support throughout the period of this research work. He has provided me with all the necessary resources including motivation and research environment without which it would not have been possible to complete this work. It was a great opportunity for me to do this work under his supervision.

I am thankful to the authors whose work I have consulted and quoted in this work.

I lack words to express my cordial thanks to all my friends for their useful comments and constructive suggestions during all the phases of my life.

Finally, I convey deep sense of gratitude towards my family members for their moral and financial support and encouragement without which it would not have been possible to bring out this thesis

Sharanjit Kaur

(801032022)

ABSTRACT

Wireless Sensor Networks consist of sensor nodes and few powerful control mobile laptops performing activities like routing, data aggregation etc over wireless media. These types of networks are becoming very popular. Various potential applications include burglar alarms, inventory control, medical monitoring and emergency response , monitoring remote or inhospitable habitats, target tracking in battlefields, disaster.

However such kind of environment prone great security threats due to the broadcast nature of the transmission medium. Furthermore, wireless sensor networks have an additional vulnerability because nodes are often placed in a hostile or dangerous environment where they are not physically protected. Sensor networks pose unique challenges, traditional security techniques used in traditional networks cannot be applied directly.

In traditional network system, Public Key Cryptography and Symmetric key Cryptography are used for providing these security services and protocols and they are used together for secure network implementation. As far as past research is concerned Symmetric key cryptography could not be applied because of resource constraints on sensor nodes particularly because of low battery life. So Asymmetric key cryptography is preferred in these type of environments.

Therefore this thesis investigates cryptography algorithms to showcase energy analysis. The work is implemented over a network created with the help of Java Sun SPOT kit, consisting of sensor node and basestation.

List of Figures

Figure 1.1 Architecture of WSN……2

Figure 1.2 Sybil Attack……..5

Figure 1.3 Wormhole Attack……..5

Figure 1.4 Cryptographic Method……..9

Figure 1.5 Symmetric Key Mechanism…….10

Figure 1.6 Diffie-Hellman protocol based on ECC……...12

Figure 2.1 Key Agreement……15

Figure 2.2 Mechanism of Certificate…….17

Figure 2.3 Certificate Validation Process……..17

Figure 2.4 Addition of two points when P=Q…….…...21

Figure 2.5 Point Addition when P≠Q…….22

Figure 2.6 Point Doubling y coordinate is non zero……...23

Figure 2.7 Point Doubling y coordinate is zero……..23

Figure 2.8 Message flow in SSL handshake……...26

Figure 2.9 RSA based handshake……....27

Figure 2.10 EC based handshake……..28

Figure 2.11 Sun SPOT Side View……29

Figure 2.12 Sun Spot Lid Open……...29

Figure 2.13 View of SunSpot……...30

Figure 4.1 Sun Spot Manager Interface…….33

Figure 4.2 Comparison of Security level of RSA/DSA and ECC…….36

Figure 4.3 Keytool command to generate RSA key……..38

Figure 4.4 List keys in keystore……..39

Figure 4.5 ECC certificate…….41

Figure 4.6 Keystore description…….42

Figure 4.7 Java code running on Sun Spot……43

Figure 4.8 Server Code……..43

Figure 4.9 Sunspot Properties File…….44

Figure 4.10 Code Deployed on Sun Spot…….45

Figure 4.11 Output of Ant listtrustedkeys command……...46

Figure 4.12 Details of ECC key in Sun Spot……....47

Figure 4.13 Details of RSA key in Sun Spot……47

Figure 4.14 Sample of RSA handshake……49

Figure 4.15 Sample of ECC handshake……49

Figure 4.16 Output of ant socket-proxy-gui…….50

Figure 4.17 Output using RSA algorithm……51

Figure 4.18 Output using EC algorithm……...52

List of Tables

Table 2.1 Power usage of typical Sun Spot…….30

Table 4.1 Computationally equivalent key sizes…….35

Table 4.2 Energy Cost of digital signature computations……....36

Table 4.3 Energy cost of key exchange computations…….37

Table 4.4 Output recorded using RSA…….52

Table 4.5 Output recorded using EC……53

ABBREVIATIONS

WSN	Wireless Sensor Network
pH	Potential Hydrogen
WLAN	Wireless Local Area Network
WEP	Wired Equivalent Privacy
WPA	Wi-Fi Protected Access
TKIP	Temporal Key Integrity Protocol
RADIUS	Remote Authentication Dial In User Service
WPA2	Wi-Fi Protected Access2
AES	Advanced Encryption Standard
3DES	Triple Data Encryption Standard
RSA	Ron Rivest, Adi Shamir and Leonard Adleman
ECC	Elliptic Curve Cryptography
ECDLP	Elliptic Curve Discrete Logarithm Problem
SSL	Secure Socket Layers
TCP	Transmission Control Protocol
RC4	Ron's Code 4
SHA	Secure Hash Algorithm
ECDH	Elliptic curve Diffie–Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
USB	Universal Serial Bus
TLS	Transport Layer Security

Table of Contents

Certificate……i

Acknowledgement ……ii

Abstract ……iii

List of Figures …….. ……iv

List of Tables ……...vi

Abbreviations……vii

Table of Contents……..viii

1 Introduction……. ……. 1

1.1 Wireless Sensor Network……1

1.1.1 Architecture of WSN ……1

1.1.2 Application area of WSN …….. ……2

1.2 Security in WSN……..3

1.2.1 Wireless Sensor Network Security Challenges…….3

1.2.2 Types of Attack……4

1.2.3 Security Requirements in Wireless Sensor Network……6

1.2.4 Wireless LAN Protocols……...7

1.2.5 Cryptography……9

2 Literature Survey……13

2.1 Wireless Sensor Network……13

2.2 Cryptography……14

2.2.1 Public Key Cryptography …….. ……...15

2.2.2 Diffie-Hellman Key Exchange ……..18

2.2.3 RSA Algorithm……...19

2.2.4 DSA Algorithm …….19

2.2.5 Elliptic Curve Cryptography……..20

2.3 SSL Operation……..25

2.3.1 RSA based Handshake……...26

2.3.2 ECC based Handshake……...27

2.4 SunSpot Motes ……28

3 Problem Statement …… 32

4 Implementation and Results……...33

4.1 Software's used in Experiment ……
33

4.1.1 Sun Spot Manager…….33

4.1.2 Java Development Kit……...34

4.1.3 ANT……...34

4.1.4 Netbeans……34

4.1.5 Keytool……..34

4.1.6 OpenSSL……...…35

4.3 Comparison of RSA and ECC……35

4.3 Experiment Details……..37

4.3.1 Create RSA/ECC certificate……..37

4.3.2 Experiment Setup and Explanation……...42

4.4 Results…….51

5 Conclusion ……...……54

References……...55

List of Publications…….59

1. INTRODUCTION

1.1 Wireless Sensor Network

Sensor networks refer to a heterogeneous system combining tiny sensors and actuators with general-purpose computing elements. These networks will consist of hundreds or thousands of self-organizing, low-power, low-cost wireless nodes deployed to monitor and affect the environment [1]. Sensor networks are typically characterized by limited power supplies, low bandwidth, small memory sizes and limited energy. This leads to a very demanding environment to provide security. The concept of wireless sensor networks is based on a simple equation [2]:

Sensing + CPU + Radio = Thousands of potential applications

Various Companies come into play for developing sensor device like MICA, INTEL and the latest is Oracle (Sun Microsystems) supporting different operating system like TinyOs, Java Squawk.

1.1.1 Architecture of WSN

WSN consists of sensor nodes which communicate via wireless media (radio) and base station (connected to host machine) which collect all the information and broadcast it further to gateway which then send to server and display it over the screen of the web client requesting the particular information. Figure 1.1 shows the basic architecture where sensor nodes are connected via radio to basestation which is further connected to host computer using wired medium. Sensor nodes send data to computer via basetation.

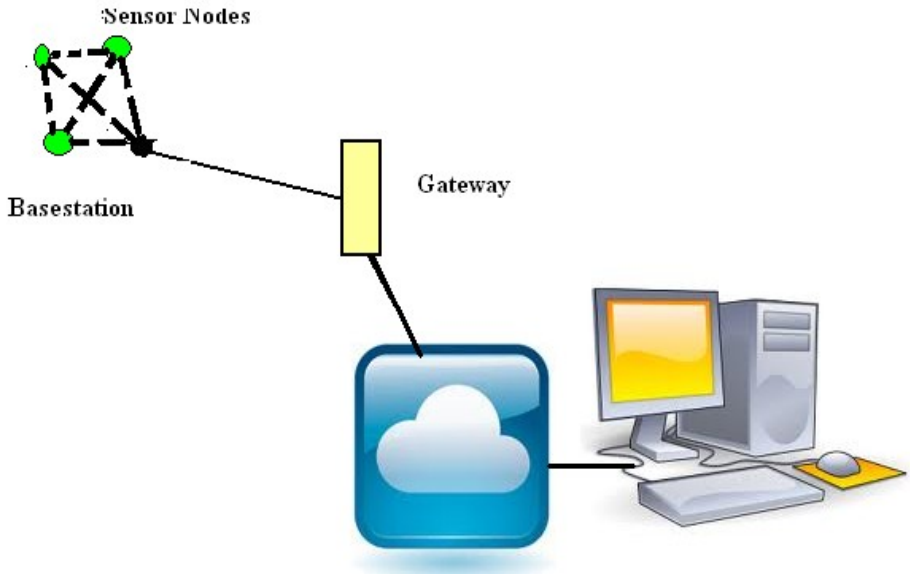


Figure 1.1 Architecture of WSN

1.1.2 Application area of WSN

Because of their small size and easy to use WSN are making may into various fields. They are used in very sensitive environment. Some of applications are listed below:

- **Military:** Wireless sensor networks are integral part of military command, control, communications, computing, intelligence, surveillance, reconnaissance and targeting systems. In battlefield environment sensor devices provide various services like battlefield assessment, targeting etc.
- **Environmental Applications:** Sensor networks are applied in habitat monitoring, agriculture research, fire detection and traffic control. WSN helps a lot in detecting soil moisture, presence of pesticides and pH level of water and help farmers to use effective measure to cultivate and save crops.
- **Health Applications:** Sensor networks are used in hospitals and health care centers. Sensor network monitor patient physiological data, control drug administration track and monitor patients and doctors inside a hospital. they help in detecting presence of unwanted material in body like stones etc.
- **Home Applications:** Research is being done to create smart environment in home. When a person enters house sensors will detect owners presence and automatically switch on AC. Environment is changed according to owners

mood. When owner sit of sofa sensor under cushion detect weight and will automatically switch on TV. When someone forcefully try to open windows vibratory sensor detect and automatically calls police.

1.2 Security in WSN

Wireless networks are vulnerable to security attacks due to the broadcast nature of the transmission medium. Furthermore, wireless sensor networks have an additional vulnerability because nodes are often placed in a hostile or dangerous environment where they are not physically protected.

1.2.1 Wireless Sensor Network Security Challenges

Protecting wireless sensor networks in all is critical. Because sensor networks pose unique challenges, traditional security techniques used in traditional networks cannot be applied directly. First, to make sensor networks economically profitable as sensor devices are limited in their energy, computation, and communication capabilities. Second, sensor nodes are often deployed in accessible areas, presenting the added risk of physical attack [3]. And third, sensor networks interact closely with their physical environments and with people, posing new security problems. Existing security mechanisms are inadequate, and new ideas are needed. A WSN presents significant challenges in designing security schemes. Five of the most important challenges are described below.

- **Wireless Medium**: Wireless medium is less secure because its broadcast nature makes eavesdropping simple. Any transmission can easily be intercepted, altered, or replayed by an adversary. It allows an attacker to easily intercept valid packets and easily inject malicious ones.
- **Ad-Hoc Deployment**: Network topology keeps changing due to node failure, addition & mobility. So nothing is known of the topology prior to deployment. Security schemes require robust designs to cope and operate in dynamic and ever changing environment.

- **Hostile Environment**: The highly hostile environment represents a serious challenge. Attackers can capture a node, physically disassemble it, and extract from it valuable information.
- **Resource Limitation**: Energy is the most precious resource for sensor networks.. Security mechanisms must be energy efficient otherwise it could increase its cost and also make decrease its usability.
- **Big Scale Network**: The high scale of sensor networks poses a significant challenge for security mechanisms. Providing security for it is equally challenging. Security mechanisms must be scalable to very large networks maintaining high computation and communication efficiency.

1.2.2 Types of Attack

- **Passive Information Gathering:** An intruder with an appropriately powerful receiver and well designed antenna can easily pick off the data stream. Interception of the messages containing the physical locations of sensor nodes allows an attacker to locate the nodes and destroy them. Even adversary can keep eye on the application specific content of messages including message IDs, timestamps and other fields. To minimize the threats of passive information gathering, strong encryption techniques needs to be used [4].
- **False Node and malicious data:** An intruder can add false node to the system and can prevent passage of valid data. Inserting malicious data is one of dangerous attacks that can occur. Malicious code injected into network could spread to all nodes, destroying whole network. Two things can happen either this sensor network can send false observation about details to legitimate user or send observation about area to malicious user.
- **The Sybil attack:** In this attack single node presents multiple identities to other nodes in the network. That means adversary can be present at more than one place at one time as single node presents multiple identities to other nodes in the network. Figure 1.2 shows Sybil attack where Sybil node has multiple identity and carries identity as A,B and C.

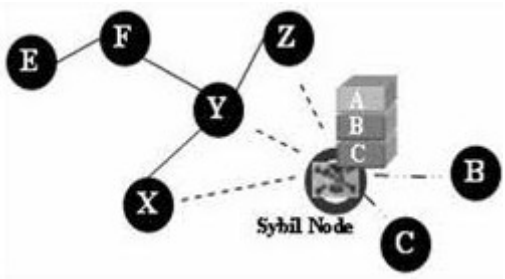


Figure 1.2 Sybil Attack

Authentication and Encryption technique can prevent any outsider but not insider from launching Sybil attack.

- **Sinkhole attacks:** In this intruder attracts network traffic by advertising itself as having shortest path to the base station. For example an intruder using a wireless-enabled laptop will have much higher computation and communication power than a normal sensor node, and it could have a high-quality single-hop link to the base station . It can then advertise imitated routing messages about the high quality route, thus spoofing the surrounding nodes to create a sinkhole (SH).
- **Wormholes:** In this attack [5] attacker can make far apart nodes to believe that they are immediate neighbors. For launching a wormhole attack, an adversary connects two distant points in the network using a direct low-latency communication link called as the wormhole link.

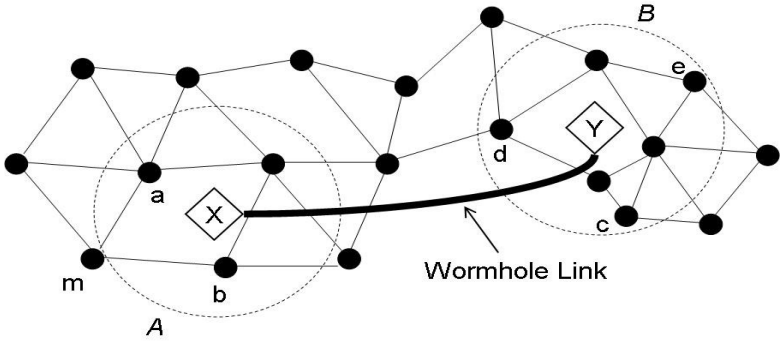


Figure 1.3 Wormhole Attack [6]

As shown in Figure 1.3 X and Y are the two end-points of the wormhole link (called as wormholes) [6]. X replays in its neighborhood (in area A) everything that Y hears in its own neighborhood (area B) and vice versa. The

net effect of such an attack is that all the nodes in area A assume that nodes in area B are their neighbors and vice versa. This, as a result, affects routing and other connectivity based protocols in the network. Once the new routes are established and the traffic in the network starts using the X-Y shortcut, the wormhole nodes can start dropping packets and cause network disruption. They can also spy on the packets going through and use the large amount of collected information to break any network security.

- **Selective Forwarding:** In this malicious node may drop packet or forward only certain packets. Adversary can drop packets from selective node and forward packets from other nodes so that nobody suspect this node.
- **Hello Flood:** Many protocols require nodes to broadcast HELLO packets to announce themselves to their neighbors, and a node receiving such a packet may assume that it is within range of the sender [7]. Attacker broadcast information and convince every node in network that adversary is its neighbor and begin exchanging information.

1.2.3 Security Requirements in Wireless Sensor Network

Major security issues for Wireless Sensor Network are listed below:-

- **Data Confidentiality:** Confidentiality means keeping information secret from unauthorized parties. A sensor network should not leak sensor readings to neighboring networks. In many applications nodes communicate highly sensitive data. Simple method to keep sensitive data secret is to encrypt the data with a secret key that only legitimate receivers possess. Public-key cryptography is little expensive to be used in the resource constrained sensor networks
- **Data Authenticity:** One can easily inject malicious data in sensor network. So receiver should make sure that data it received is correct and legitimate data as in this malicious data can lead to wrong interpretation by receiver which can have very bad impact as in critical decision making process wrong information can lead to catastrophic results also.

- **Data Integrity:** Data integrity ensures the receiver that the received data is not change by an adversary during transfer. Data Authentication can provide Data Integrity also.
- **Data Freshness:** Data freshness implies that the data is recent, and it ensures no old messages are replayed over network. For this counter must be used that can determine freshness.
- **Robustness and Survivability:** The sensor network should be robust against various security attacks, and if an attack succeeds, its impact should be minimized. The compromise of a single node should not break the security of the entire network.
- **Availability:** Availability ensures that services and information can be accessed at the time they are required. In sensor networks there are many risks that could result in loss of availability such as sensor node capturing and denial of service attacks.
- **Secure Management:** Management is required in every system that is constituted of multi components, and handles sensitive information. In the case of sensor networks secure management is needed on base station level; since sensor nodes communication ends up at the base station, issues like key distribution to sensor nodes in order to establish encryption and routing information need secure management.
- **Quality of Service:** Assuring the quality of Service objective is a big challenge to security designers. As sensor networks have several limitations (e.g. energy, processing and memory capacities etc.), the achievement of quality of service becomes even more constrained.

1.2.4 Wireless LAN Protocols

Wireless LAN is wireless local area network in which two or more computing devices are connected without wire. WLAN utilizes spread-spectrum technology based on radio waves to enable communication between devices in a limited area. For security of WLAN many protocols are defined. Brief description about these protocols is discussed below.

WEP (Wired Equivalent Privacy)

The WEP algorithm protects wireless communication from eavesdropping & prevents unauthorized access to a wireless network. WEP relies on RC4 secret key cryptographic algorithm. The secret key is shared between communicating parties, clients & access points. In the WEP encryption algorithm, payloads are basically encrypted via the RC4 cipher with keys of 64 or 128 bits. The WEP algorithm inserts two main overhead fields, namely Initialization Vector (IV) and Integrity Check Value (ICV). Here, IV is a randomly-selected 24-bit integer value that will be combined with the user's secret key of 56 or 104 bits to form a 64-bit (or 128-bit) encryption key for each packet [8]. On the other hand, ICV is the 32-bit checksum computed via CRC-32 for each packet of payload. The WEP encryption algorithm pads and transmits the 24-bit IV and 32-bit ICV information as the header and trailer together with each packet of encrypted payload.

Weaknesses in WEP

1. **Key management & key size:** Key management is not specified in WEP standard. So without key management, keys will be long lived & of poor quality. Most of WEP users have one single WEP key shared between every node on the network. Synchronizing the change of keys is tedious and difficult.
2. **Authentication messages can be easily forged:** Attacker just need to know shared WEP key. He can determine RC4 stream used to encrypt response and use that stream to encrypt any challenge attacker receives.

WPA (Wi-Fi Protected Access)

WPA was developed in response to the shortcomings of WEP. One of the key technologies behind WPA is the Temporal Key Integrity Protocol (TKIP). TKIP means that the Encryption is mandatory under WPA. In the past standard it was optional. TKIP manages encryption keys and the synchronization of changing keys across the wireless network. Together with 802.1X/EAP authentication, TKIP employs a key hierarchy that greatly enhances protection. It also adds a Message Integrity Check (MIC, sometimes called "Michael") to protect packet from being tempered. WPA operates in either WPA-PSK mode (Pre-Shared Key or WPA-

Personal) or WPA-802.1x mode (RADIUS or WPA-Enterprise). In the Personal mode, a pre-shared key is used for authentication. In the Enterprise mode, which is more difficult to configure, the 802.1x Remote Authentication Dial In User Service (RADIUS) servers and an Extensible Authentication Protocol (EAP) are used for authentication.

WPA2 (Wi-Fi Protected Access2)

WPA2 uses Advanced Encryption Standard (AES) instead of Temporal Key Integrity Protocol (TKIP) to provide stronger encryption mechanism. AES supports 128-bit, 192-bit and 256-bit keys. There are two versions of WPA2: WPA2-Personal, and WPA2-Enterprise. WPA2- Enterprise verifies network users through a server.

WLAN protocols aim is to provide authentication, confidentiality and data integrity. Data privacy is achieved by cryptography which is discussed below.

1.2.5 Cryptography

Cryptography is basically the conversion of data into a secret code for transmission over a public network. Cryptography can be divided into transposition and substitution [9]. Transposition is rearrangement of letters of a message according to a certain algorithm. In Substitution characters in the original text or plain text, are substituted by other characters or symbols using certain algorithm . The original text or plaintext is turned into a coded equivalent called cipher text via an encryption algorithm. The cipher text is decrypted at the receiving end and turned back into plaintext. It involves two mechanism Encryption and Decryption. Figure1.4 shows the cryptographic mechanism.

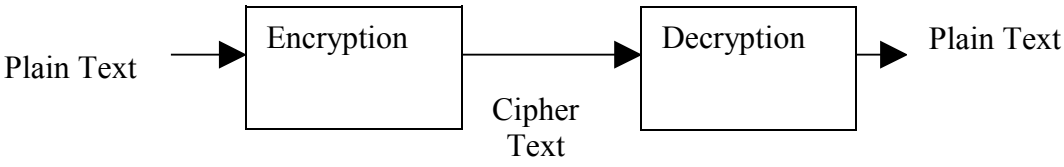


Figure 1.4 Cryptographic Method

Using same Algorithm data is encoded on sender side and decode on receiver side. The algorithm used are very complex as mathematical formulae are used, so that intruder is not easily able to read or modify the original data. Purpose of cryptographic algorithms is as follows.

- Authentication: The process of proving one's identity.
- Confidentiality: Ensuring that no one can read the message except the intended receiver.
- Integrity: Assuring the receiver that the received message has not been altered in any way from the original.
- Non-repudiation: A mechanism to prove that the sender really sent this message.

Cryptographic algorithms are divided into two kind Symmetric and Asymmetric Algorithms. They are briefly discussed below.

Symmetric Algorithm

A single key is used for both encryption and decryption. As shown in Figure 1.5, on encryption the data is encrypted by any encryption algorithm using the key [10]. Only the user having the access to the same 'key' can decrypt the encrypted data. This method is known as private key or symmetric key cryptography. There are several standard symmetric key algorithms defined. Examples are AES, 3DES etc.

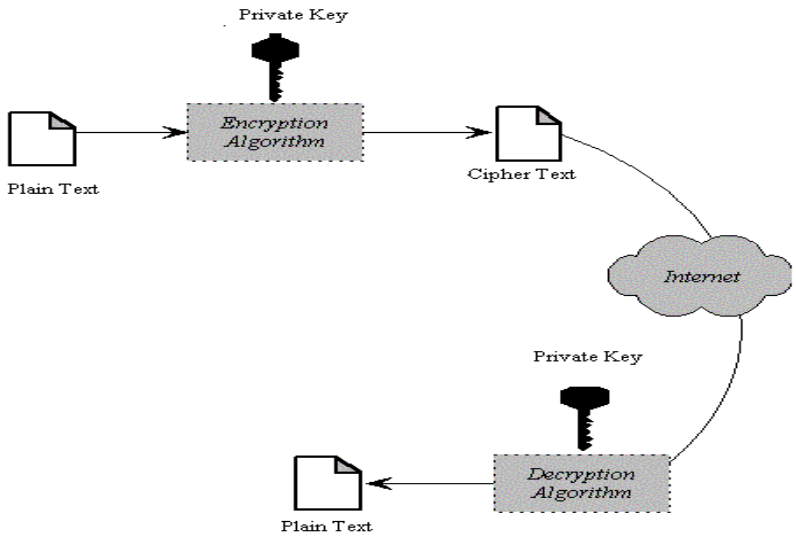


Figure 1.5 Symmetric Key Mechanism

There are various issues with this algorithm like key distribution, communicating nodes use same keys and keys transferred over the media any intruder can get key and decrypt the message thus reducing security. No authentication and repudiation is provided using symmetric algorithms.

Asymmetric Algorithm

In public key cryptography each user or the device taking part in the communication have a pair of keys, a public key and a private key, and a set of operations associated with the keys to do the cryptographic operations . Only the particular device knows the private key whereas the public key is distributed to all devices taking part in the communication. It is discussed in detail in next chapter.

Limitations

Sensor networks are vulnerable to resource consumption attacks. Intruder can repeatedly send packets to drain the nodes batteries and waste network bandwidth. Computing power, memory, and battery life of devices are more constrained in WSN and all above security protocols use lots of memory and require lot of battery power to successfully implement them. Implementing them in wireless Sensor network actually degrades performance of WSN due to large battery and memory required by them to implement. Elliptical curve cryptography provides same level of security with less parameters required thus improving the performance of Wireless Sensor Network.

Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a public key cryptography. ECC is found to be best solution till now for Wireless Sensor networks than RSA as RSA does not satisfy the resource constraints of WSN. The original Diffie-Hellman algorithm with RSA requires a key of 1024 bits to achieve sufficient security but Diffie-Hellman based on ECC can achieve the same security level with only 160 bit key size [11]. Although in today's world bit size has to be increased to increase security as adversaries have got

hold of powerful computing devices. Elliptic Curve Diffie-Hellman scheme works as shown in the Figure 1.6

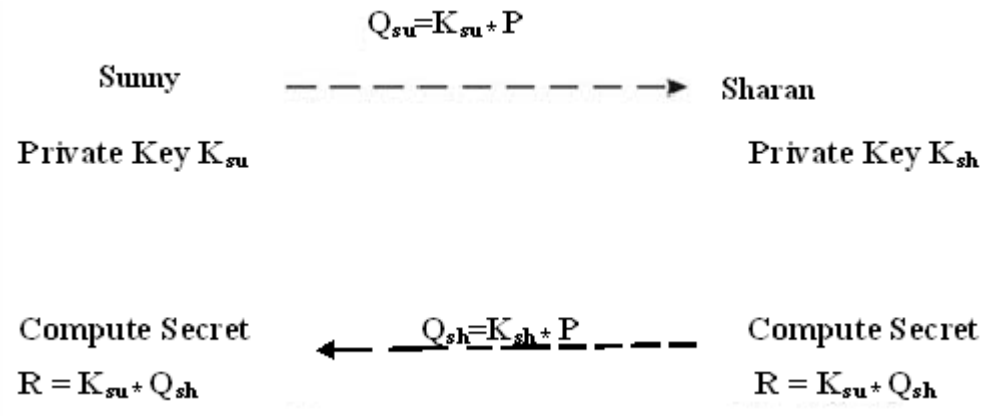


Figure 1.6 Diffie-Hellman protocol based on ECC

Sunny and Sharan agree on a particular curve with base point *P*. They generate their public keys by multiplying with their private keys namely K_{su} and $K_{sh.}$ After sharing public keys, they generate a shared secret key by multiplying public keys by their private keys. The secret key is $R= K_{su} * Q_{sh}= K_{sh} * Q_{su}$. With the known values of Q_{su}, Q_{sh} and *P* it is computationally very difficult for an eavesdropper to calculate K_{su} and K_{sh} which are the private keys of Sunny and Sharan. As a result, intruder cannot figure out which is the shared secret key.

This thesis is an attempt to implement ECC and RSA and compare their suitability for WSN. Next chapter is literature survey which discusses in detail work done by other researchers.

2. LITERATURE SURVEY

Brief introduction of Wireless Sensor Network is given in previous chapter. Researchers till date have done numerous studies on WSN and pointed out various issues and gave new concepts to improve security in WSN. As usage of WSN is growing need to secure it is also growing and researchers try to understand concepts in detail and devise new ways so that WSN can be improved. The following sections describe the research till date in brief by different authors.

2.1 Wireless Sensor Network

The study of wireless sensor networks is challenging in that it requires an enormous breadth of knowledge from an enormous variety of disciplines. These networks can consist of hundreds or thousands of self organizing, low power, low cost wireless nodes deployed en masse to monitor and affect the environment. Potential applications include burglar alarms, inventory control, medical monitoring and emergency response , monitoring remote or inhospitable habitats, target tracking in battlefields, disaster relief networks, early fire detection in forests, and environmental monitoring. They can be used in virtually any environment, even those where wired connections are not possible, where the terrain is inhospitable, or where physical placement is difficult.

All security threats, risks involved in Wireless Sensor network and various challenges which this type of environment poses are all discussed in previous chapter. There are many ways devised by researchers till date to try and protect the data transferred in these environment without compromising energy, taking less time and less memory. Cryptography is the technique which is used to protect, authenticate data while data is transferred in unsecured WSN environment which is discussed ahead in detail.

2.2 Cryptography

Today's cryptography is more than encryption and decryption. Cryptography is the study of “mathematical” systems for solving two kinds of security problems: privacy and authentication [12]. A privacy system prevents the intruder from getting hold of the message which is being transmitted over public channel, thus assuring the sender of a message that it is being read only by the intended recipient. An authentication system prevents the unauthorized injection of messages into a public channel, assuring the receiver of a message of the legitimacy of its sender. Cryptography provides mechanisms for such procedures. A digital signature binds a document to the owner of a particular key, while a digital timestamp binds a document to its creation at a particular time[13].

Key management is area of major problem in cryptography. Key management is the set of techniques and procedures supporting the establishment and maintenance of keying relationships between authorized parties. As the entire operation is dependent upon the security of the keys, it is sometimes appropriate to devise a complex mechanism to manage them. The two components required to encrypt data are an algorithm and a key. As algorithm is generally known to everyone and the key is kept secret. As already discussed in previous chapter, in symmetric encryption algorithms the same key is used for encryption and decryption but it has its own limitations. Two Stanford University researchers, Whitfield Diffie and Martin Hellman, wrote a landmark paper, “New Directions in Cryptography,” in 1976 [12]. The paper suggested that perhaps encryption and decryption could be done with a pair of different keys rather than with the same key. The decryption key would still have to be kept secret, but the encryption key could be made public without compromising the security of the decryption key. This concept is called public-key cryptography because the encryption key could be made known to anyone. Since in experiment different public key algorithms are analyzed and implemented so Public Key cryptography is discussed in which detail of key agreement, encryption technique and digital signature is discussed which helps to understand experiment done.

2.2.1 Public Key Cryptography

In public key cryptography each device taking part in the communication have a pair of keys, a public key and a private key, and a set of operations associated with the keys. Only the particular device knows the private key whereas the public key is distributed to all other devices taking part in the communication. Since the public key does not compromise the security of the algorithms, it can be easily exchanged. Secured link can be established between two communicating parties by exchanging only public keys and public constants if any. Any intruder who has access only to the exchanged public information, will not be able to read and interpret data communicating between secured link unless it has access to the private key of any of the communicating parties.

Key Agreement

Key agreement is a method in which the device communicating in the network establishes a shared secret between them without exchanging any secret data. In this method the devices that need to establish shared secret (session key) between them exchange their public keys [10]. Both the devices on receiving the other device's public key performs key generation operation using its private key to obtain the shared secret. Examples of key agreement algorithms are DH, RSA and ECDH.

Consider two devices A and B. Let P_A and $PU_B(P_A , C)$ be the private key and public key of device A, and P_B and $PU_B (P_B, C)$ be the private key and public key of device B respectively. Both device exchanges their public keys as shown in Figure 2.1

Device A, having got the public key of B, uses its private key to calculate shared secret $KA=Generate\_Key(P_A, PU_B (P_B , C))$

Device B, having got the public key of A, uses its private key to calculate the shared secret $KB=Generate\_Key(P_B, PU_A (P_A, C))$

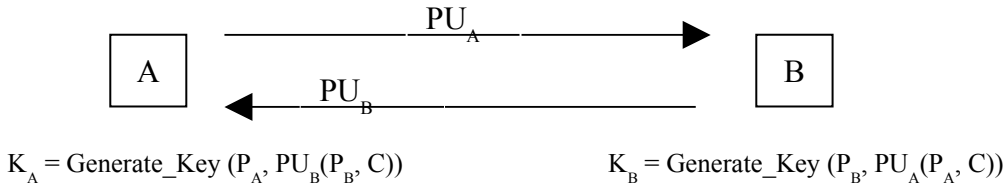


Figure 2.1 Key Agreement [10]

The key generation algorithm will generate keys at the device A and B which will be the same, that is shared secret $K_A=K_B=K(P_A, P_B, C)$. Any intruder having access only to the public keys $PU_A(P_A, C)$ and $PU_B(P_B, C)$, will never be able to obtain the shared secret K. Examples of key agreement algorithms are DH, RSA and ECDH. For establishing shared secret it is important that device A receives the correct public key from device B and vice versa. Digital Certificate helps to deliver the public key in authenticated method.

Encryption

Encryption is a process in which the sender encrypts the message in such a way that only the recipient will be able to decrypt the message. Consider a device B whose private key and public key are P_B and PU_B respectively. Since PU_B is public key all devices will be able to get it. For any device that needs to send the message in a secured way to device B, it will encrypt the data using B's public key. The encrypted message can only be decrypted using B's private key. On receiving the message the B decrypts it using its private key P_B. Since only B knows its private key P_B it can only decrypt the message [10].

Digital Signature

To ensure authenticity of the message method used is Digital signature. In this a message can be signed by a device using its private key. Any device that has got the access to the public key of the signed device can verify the signature. Thus the device receiving the message can ensure that the message is actually signed by the legitimate device and is not modified during the transfer. If any of the data or signature is modified, the signature verification fails. The examples of Digital Signature algorithms are RSA, DSA and ECDSA [10].

Certificate: There is authority which is trusted by all devices in a network for data transfer called Trusted Certificate Authority (CA) which signs the public keys and the unique identifiers of all devices. These signed data (public key, IDs etc.) along with the signature arranged in a standard format is called as the certificate [14] . A standard

digital certificate format is X.509 certificates. All the devices that take part in secured and trusted communication have to obtain a certificate from the trusted authority. Devices exchanges their respective certificate instead of public key.

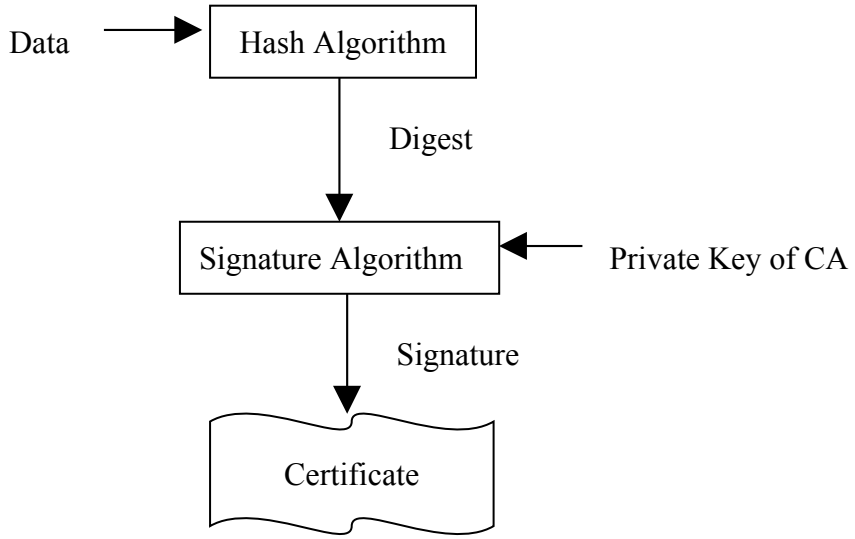


Figure 2.2 Mechanism of Certificate [10]

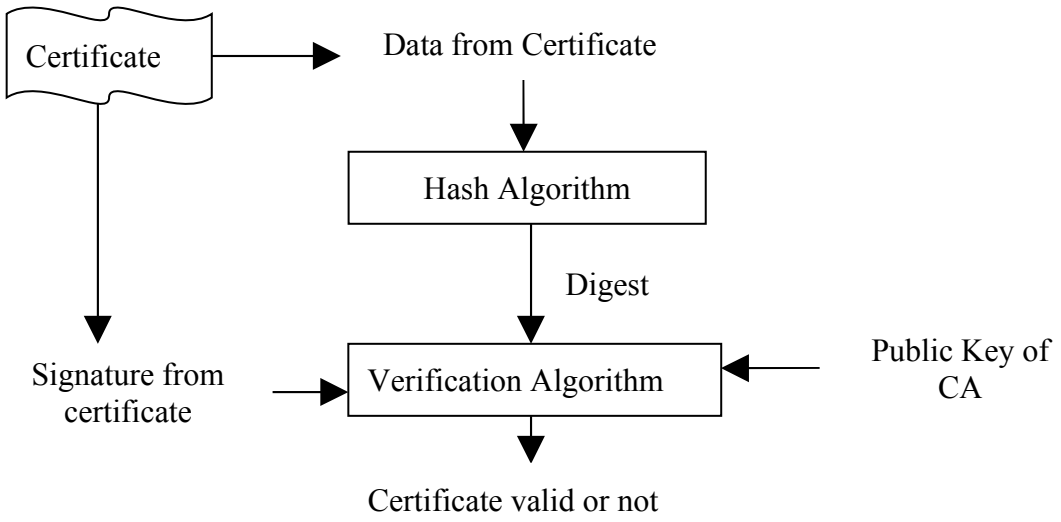


Figure 2.3 Certificate Validation Process [10]

Figure 2.2 show mechanism of certificate creation. The CA first finds the digest or information of the device data and CA specific data using a hash algorithm. CA then

signs the hash using its private key and combines the data and signature On receiving certificate device extract the information from the certificate, checks the ID and other data in the certificate. The signature in the certificate is verified using CA's public key. Figure 2.3 show certificate validation process. The public keys are generally stored as certificates and the root CA public key is stored as self signed certificate. As the number of devices are increasing and the location of these devices is distributed over different parts of the world, a central certificate authority is not sufficient to issue and maintain the certificate of all the devices. Best solution for this is Certificate hierarchy. Various different algorithms to transfer data securely over WSN network mainly using above described method are discussed below.

2.2.2 Diffie-Hellman Key Exchange

The key exchange works over a public channel in this way [14]:

Suppose that Alice and Bob would like to communicate securely. They agree on a large prime p and a base b. It is preferable for security, though not necessary, that b be a generator with respect to p. Otherwise, the pool of possible keys is reduced, leaving the system more vulnerable to attack [15].

1. Privately, Alice selects some X_i between 2 and p - 2 and Bob selects some X_j . They never share these numbers.
2. Alice computes $Yi = b^{X_i} \bmod p$ and Bob computes $Yj = b^{X_j} \bmod p$. They exchange these numbers across the public channel.
3. Alice calculates $Y_j^{X_i} \bmod p = b^{X_j X_i} \bmod p = K_{ij}$ and Bob finds the same K_{ij} by calculating $Y_i^{X_j} \bmod p = b^{X_i X_j} \bmod p = K_{ij}$. This is their shared key for a symmetric cipher.

Any intruder who is watching can see p, b, Y_i and Y_j but cannot find X_i, X_j and K_{ij}. unless intruder can solve the discrete log problem, $X = \log_b Y \bmod p$, which appears to be difficult. Diffie-Hellman key exchange provides security from eavesdropper but no authentication is provided. Diffie and Hellman [3] suggest that this role be played by a trusted third party because of which Digital Signature and certificate came into picture.

2.2.3 RSA Algorithm

The RSA algorithm is named after Ron Rivest, Adi Shamir and Len Adleman, who invented it in 1977. RSA gets its security from integer factorization problem. Difficulty of factoring large numbers is the basis of security of RSA [16].

Key generation:

1. Select random prime numbers p and q, and check that p != q
2. Compute modulus n = pq
3. Compute phi, $\varphi = (p - 1)(q - 1)$
4. Select public exponent *e*, $1 < e < \varphi$ such that $\gcd(e, \varphi) = 1$
5. Compute private exponent $d = e^{-1} \bmod \varphi$
6. Public key is {n, e}, private key is d

Encryption: $c = m^e \bmod n$, decryption: $m = c^d \bmod n$

Digital signature: $s = H(m)^d \bmod n$, Verification: $m' = s^e \bmod n$, if $m' = H(m)$ signature is correct. H is a publicly known hash function.

2.2.4 Digital Signature Algorithm (DSA)

A digital signature is represented in a computer as a string of binary digits. A digital signature is computed using a set of parameters and authenticates the integrity of the signed data and the identity of the signatory. An algorithm provides the capability to generate and verify signature. Signature generation makes use of a private key to generate a digital signature [17]. Signature verification makes use of a public key, which corresponds to, but is not the same as, the private key. Each user possesses a private and public key pair. Public keys are assumed to be known to the public in general. Private keys are never shared. Anyone can verify the signature of a user by employing that user public key. Only the possessor of the user private key can perform signature generation.

2.2.5 Elliptic Curve Cryptography

Various public key cryptography algorithms are used to transfer data securely in WSN. Elliptic curve cryptography (ECC) is relatively new technology compared to other public key cryptography such as RSA. Elliptic curves were proposed for use as the basis for discrete logarithm-based cryptosystems almost 25 years ago independently by Victor Miller [18] of IBM and Neal Koblitz [19] of the University of Washington. Elliptic key operates on smaller key size. A 160-bit key in ECC is considered to be as secured as a 1024 bit key in RSA. ECC operates on the points in the elliptic curve $y^2=x^3+ax+b$, where $4a^3+27b^2\neq 0$. Equation of elliptic curve is in real coordinate. In prime field operation the elliptic curve equation is modified as:

$y^2 \bmod p = x^3+ax+b \bmod p$, where $4a^3+27b^2 \bmod p \neq 0$.

Basic Concept of ECC

An elliptic curve over a finite field F_P is composed of a finite group of points (x_i, y_i) where integer coordinates x_i, y_i satisfy the long Weierstrass Equation.

$y^3 + a_1 xy + a_3 y = x^3 + a_2x^2 + a_4x + a_6$

And the coefficients a_1, a_2, a_3, a_4, a_6 are elements of F_P and are the parameters of the curve. The curve discriminent is $\Delta\neq 0$ and there is also a point at infinity denoted by Θ. If F_P is a field of characteristics 2, then the curve is called a binary elliptic curve. Since the field is F_P is generally used is cryptographic applications, the equation of elliptic curve is simplified to [20]

$$y^2 = x^3 + ax + b, \quad a,b \in Fp, \quad 4a^3 + 27b^2 \neq 0 \qquad (2.1)$$

The requirement $4a^3 + 27b^2 \neq 0$ ensures that E is non-singular, this means in particular that one may compute the tangent in every point on the curve.

The set of rational points in E over Fp denoted by E(Fp) is

$$E(Fp) = \{(x, y) \in Fp^2 : y^2 = x^3 + ax + b\} \cup \{O\}, \qquad (2.2)$$

where O is the point at infinity. The elements of $E(F_P)$ are called points of elliptic curve defined by equation 2.1. The public key is a point in the curve and the private key is a random number.

It can be seen that every line intersecting the curve E intersects the curve in exactly three points, where:

1. point P is counted twice if line is tangent to the curve at P
2. Point at infinity is also counted when line is vertical.

ECC Arithmetic

Point Addition: A negative of a point is the reflection of that point with respect to x axis. Given appoint P, in negation P + (-P) = 0. The line connecting two points intersects the curve O. Point of infinity O is like sitting at the top of y-axis and is present on every vertical line. As shown in Figure 2.4 -P is simply reflection of P in the x axis that is if $P = (x_1, y_1)$ then $-P = (x+1, -y_1)$.

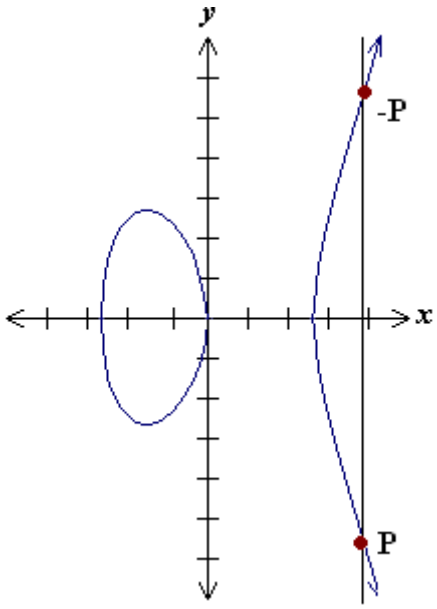


Figure 2.4 Addition of two points when P=Q [21]

For an elliptic curve E take any two points $P,Q \neq O$. Point addition is the process of adding these points to get another point R on the same elliptic curve. If P=Q draw the line tangent to E at P If $Q \neq -P$ as shown in Figure 2.5 then the line drawn through point P and Q will intersect the elliptic curve at one or more points –R.

P+Q can be found by reflecting the point –R with respect to x axis. In $E(F_p)$ the same addition operation is used but the calculations are done mod p.

Given two points *P* and *Q*, with coordinates (x_1, y_1) and(x_2, y_2) respectively, their addition results in a point R on the curve with coordinates (x_3, y_3) where x_3 and y_3 satisfy

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

Case where $x_1 \neq x_2$ and $y_1 \neq y_2$. Then points x_3 and y_3 can be found by first finding the slope of line m through P and Q and then calculating the values of x_3 and y_3.

$$m = (y_2-y_1) /(x_2-x_1) \bmod p$$

Such that

$$x_3 = m^2-x_1-x_2 \bmod p$$

$$y_3 = m(x_1-x_3)- y_1 \bmod p$$

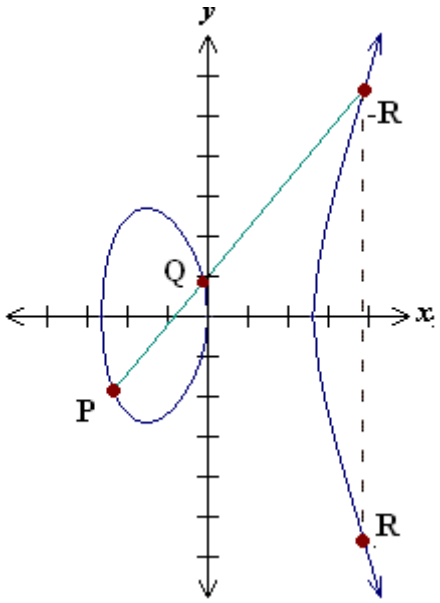


Figure 2.5 Point Addition when P≠Q [21]

Point Doubling: For an elliptic curve E take any point P ≠O. point doubling is the process of adding the point P to itself to get another point R on same elliptic curve. If the y coordinate of the point P ≠0 as shown in Figure 2.6 then the tangent line drawn

at P will intersect the elliptic curve at exactly one point –R. Graphically 2P can be found by reflecting the point –R with respect to x-axis.

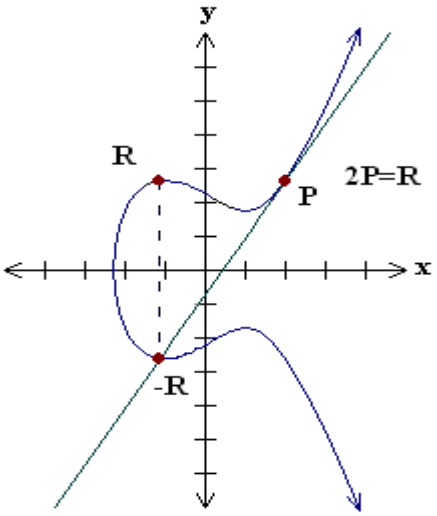


Figure 2.6 Point Doubling y coordinate is non zero [21]

Consider a point P (x_1,y_1) be in $E(F_p)$ with P $\neq$O. Considering case where $y_1 \neq 0$. Then points x_3 and y_3 can be found by first finding the slope of line m and then calculating the values of x_3 and y_3.

$$M=(3x_1 - a)/2y_1 \bmod p$$

$$x_3=m^2 -2x_1 \bmod p$$

$$y_3=m(x_1-x_3)-y_1 \bmod p$$

If the y coordinate off the point P =0 as shown in Figure 2.7 then the tangent at this point intersects the curve at O So 2P=O.

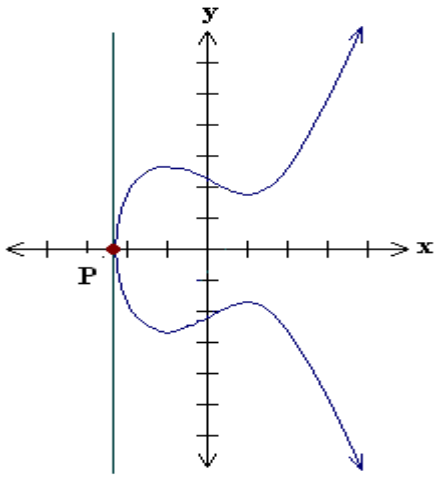


Figure 2.7 Point Doubling y coordinate is zero [21]

Scalar Point multiplication: It is process of adding the point P to itself k times to another point Q on same elliptic curve.

$$Q=kP=P+P+P+\ldots\ldots P, \qquad \text{here } P+P+\ldots P=k$$

Where $k<|EF_P|$ is a scalar. Scalar multiplication of a point on E ca be done through combination of point addition and point doubling e.g.

$$11P=2((2P)+P)+P$$

Which is called double and add algorithm for scalar multiplication.

Elliptic Curve Discrete Logarithm Problem

Security of ECC depends on the difficulty of Elliptic Curve Discrete Logarithm Problem. Given an elliptic curve E over a finite field F_p , a point on that curve, P , and another point known to be an integer multiple of that point, Q , the Elliptic Curve Discrete Logarithm Problem is to find the integer n such that nP=Q .

Problem is considered computationally difficult unless the curve has a ``bad" number of points over the given field which make it possible to ECDLP break. If the number of points on E over F_p is the same as the number of elements of F_p that is $|EF_p|= p$, then the curve is vulnerable [22].

ECC Domain Parameters

There are other parameters besides a and b parameters of curve that must be agreed by both parties involved in secured connection using ECC called domain parameters. Domain parameters for EC over F_p are p, a, b, G, n and h. p is the prime number defined over F_p. a and b are the parameters defining the curve $y^2= x^3+ax+b \bmod p$. G is the generator point (x_G,y_G) a point on the elliptic curve chosen for cryptographic operations. N is the order of elliptic curve. The scalar for point multiplication is chosen as number between 0 and n-1. h is the cofactor where $h= \#E(F_p)/n$. $\#E(F_p)$ is the number of points on an elliptic curve [22].

2.3 SSL Operation

Secure Sockets Layer is the most widely deployed and used security protocol on the Internet today. Whatever data is exchanged over insecure public network SSL offers encryption, source authentication and integrity protection. It operates above a reliable transport service like TCP and has the flexibility to accommodate different cryptographic algorithms for key agreement, encryption and hashing. SSL employ a public-key cryptosystem for authentication and to derive shared secret keys. These keys are then used in fast symmetric-key and hashing algorithms to ensure confidentiality, integrity and source authentication of bulk data. Particular combinations of these algorithms is called cipher suite. For example, a cipher suite such as RSA-RC4-SHA would indicate RSA as the key exchange mechanism, RC4 for bulk encryption, and SHA for hashing [23].

The two main components of SSL are the Handshake protocol and the Record Layer protocol. The Handshake protocol allows an SSL client and server to negotiate a common cipher suite, authenticate each other, and establish shared secret using public-key cryptographic algorithms. The Record Layer derives symmetric-keys from the shared secret and uses them with faster symmetric-key algorithms for bulk encryption and authentication of application data.

In a typical SSL handshake, the client and server first exchange random nonce and negotiate a cipher suite with *ClientHello* and *ServerHello* messages. The server then sends its signed public-key either in the *ServerCertificate* message or the *ServerKeyExchange* message. To verify the server's public key, the client performs public key operation. Then the client generates the pre shared secret encrypts it with the server's public key and sends it in the *ClientKeyExchange* message. The server uses its private-key to decrypt the pre shared secret. Both end-points then use the pre shared secret to create a shared secret which, along with previously exchanged nonce, is used to derive the cipher keys, initialization vectors and MAC (Message Authentication Code) keys for bulk encryption by the Record Layer [24]. Figure 2.8 show message flow of SSL handshake.

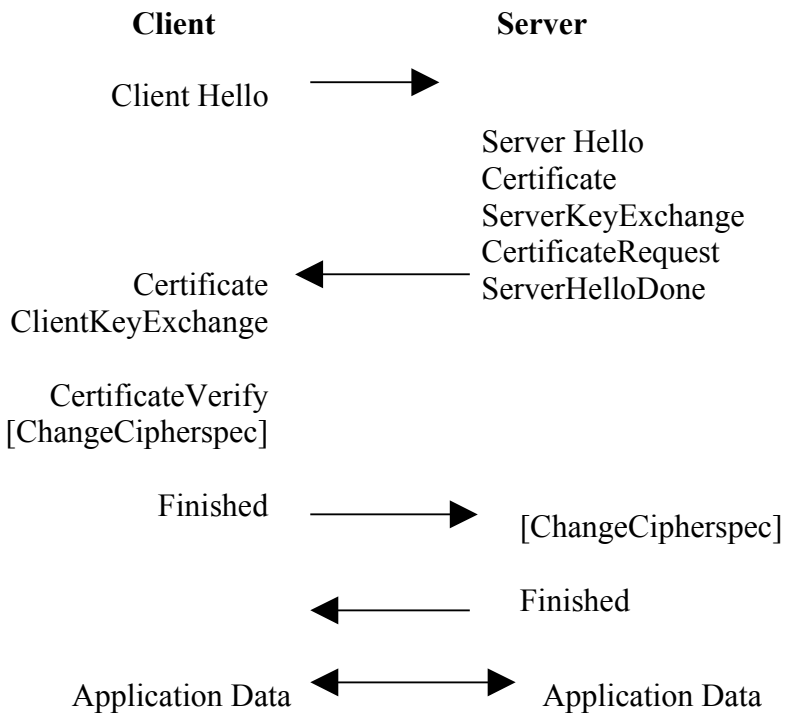


Figure 2.8 Message flow in SSL handshake [24]

2.3.1 RSA based Handshake

In this type of SSL handshake, the client and server the client and server first exchange random nonce (random number issued in an authentication protocol to ensure that old communications cannot be reused in replay attacks) C.Rand in this client indicates version of SSL it implements and negotiate a cipher suite with *ClientHello* and *ServerHello* messages. Next the server selects one of the proposed cipher suites and responds to the client with a 32-byte random value and its RSA-based certificate. The client must perform an RSA verify operation to verify the server's certificate, an RSA encrypt operation with the server's public key to encrypt a random 48-byte secret key, and an RSA sign operation on a derivate of the 32-byte random number proving possession of its private key [25].

In third message, sent from the client to the server, includes the client's certificate and a finished message, which is the first symmetrically encrypted message based on the shared secret. This message also includes the encrypted secret key and the client's signature. The server can now verify the client's identity by verifying the client's signature and prove its identity to the client by being able to decrypt the secret key

with its private key. Mutual authentication is achieved by shared secret. The server ends the handshake with a finished message containing a keyed hash of the data exchanged .

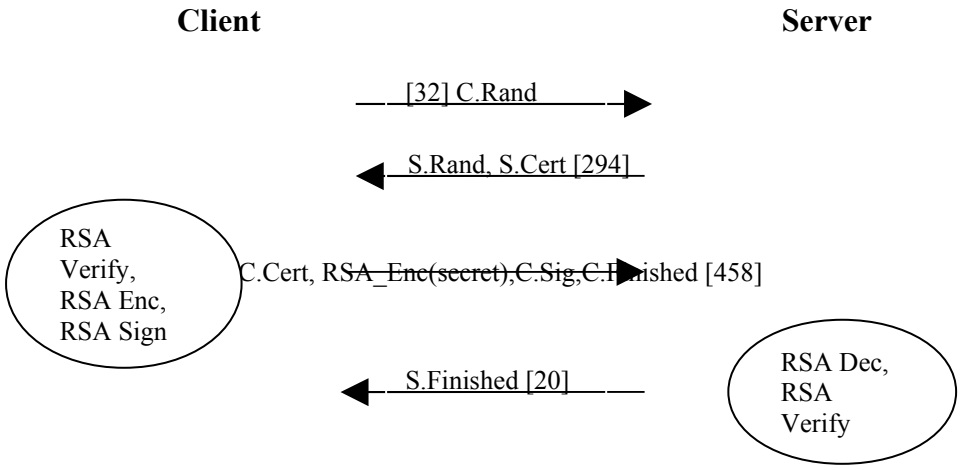


Figure 2.9 RSA based handshake. Payload message size (in bytes) shown in bracket [25].

2.3.2 ECC based Handshake

Through the first two message (processed in the same way as for RSA) the client and server negotiate an ECC based cipher suite (for example, ECDH-ECDSA-RC4-SHA). the client performs an ECDSA verify operation to verify the server's certificate and an ECDH operation to calculate the shared secret. In third message the server computes the shared secret using ECDH and performs an ECDSA operation to verify the client's certificate. Mutual authentication is achieved through mutual possession of the shared secret. the server ends the handshake with a finished message containing a keyed hash of the data exchanged so far.

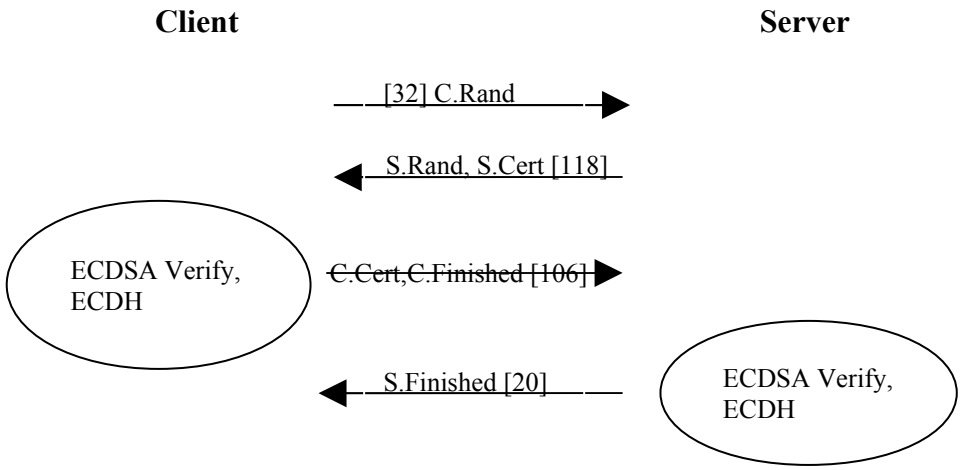


Figure 2.10 EC based handshake. Payload message size (in bytes) shown in bracket [25].

There are various wireless sensor motes and hardware structure of Sun Spot motes which is used in experiment, is described next.

2.4 SunSpot Motes

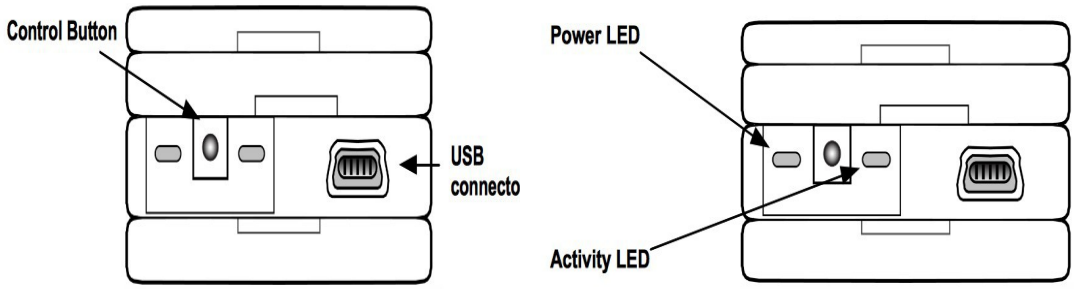
A Sun SPOT kit contains the following [26]:

- one basestation Sun SPOT unit with USB power
- two free-range Sun SPOT units with onboard battery
- a USB cable for connection between a standard USB port and a Sun SPOT unit
- one Sun SPOT software CDROM
- two mounting brackets, each allowing a Sun SPOT unit to be wall-mounted
- one mounting bracket to allow mounting of a Sun SPOT to a circuit board

Introduction

The Sun SPOT unit has one switch and one connector that are accessible without removing the case lid. These are shown below in Figure 2.11 (a). The switch is the

Sun SPOT unit control switch. If the Sun SPOT unit is off, pressing the switch turns the Sun SPOT unit on. Sun SPOT unit also has two LEDs behind the plastic casing shown in Figure 2.11 (b). The activity LED is under Java program control and can be used in applications, but it is usually used by the system software.



(a) (b)

Figure 2.11 Sun SPOT Side View [26]

When lid is removed one can see 2 switches and 8 LEDs as shown in Figure 2.12. The LEDs have red, blue, and green components. The switches and LEDs have no pre defined purpose and are under Java program control.

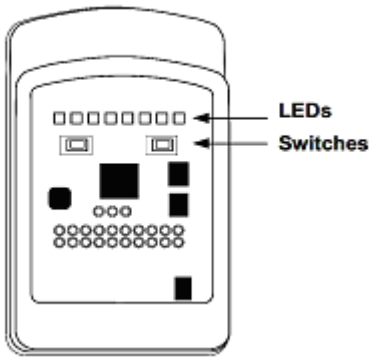


Figure 2.12 Sun Spot Lid Open [26]

The capacity of the built-in battery is 720 milliampere-hours. The drain of the system varies with use as shown in Table 2.1. Changing the transmit power of the radio effects the current draw slightly. Reducing the transmission power from 0db to -25db results in a savings of about 3 milliamperes. LEDs also consume battery and different color of LED lights can be used of which green color light draws least battery charge.

Table 2.1 Power usage of typical Sun Spot [26]

Processor Board State	Radio	Sensor Board	Current draw
Deep sleep mode	Off	Any	~33 microamperes
Shallow sleep	Off	Not Present	~24 milliamperes
Shallow sleep	On	Not Present	~40 milliamperes
Awake, actively calculating	Off	Not Present	~80 milliamperes
Awake, actively calculating	On	Not Present	~98 milliamperes
Shallow sleep	Off	Present	~31 milliamperes
Shallow sleep	On	Present	~46 milliamperes
Awake, actively calculating	Off	Present	~86 milliamperes
Awake, actively calculating	On	Present	~104 milliamperes

Configuration of the Sun SPOT platform

The eSPOT, has a main processor running the Java VM "Squawk" and which serves as an IEEE 802.15.4 wireless network node. The eSPOT has flexible power management and can draw from rechargeable battery, USB host or be externally powered [27]. Figure 2.13 show Free Sun Spot which has Sunroof, sensor board, processor board, battery.

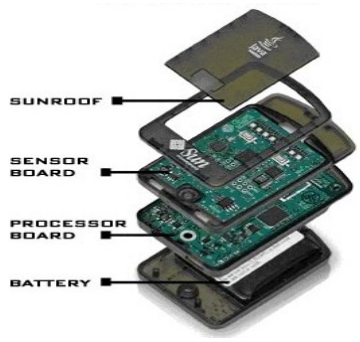


Figure 2.13 View of SunSpot

The eDemo board (sensor board) contains a 3-axis accelerometer, an ambient light sensor, eight tricolor LEDs, two push buttons, six analog input pads, four high current high voltage output pads, and five general I/O pads.

The eSPOT main board (processor board) contains the following [27]:

- main processor: The main processor is an Atmel AT91RM9200 system on a chip (SOC) integrated circuit.
- memory : consisting of a 4MByte NOR Flash memory and a 512KByte pseudo-static random access memory.
- power management circuit
- 802.15.4 radio transceiver and antenna (standard which specifies the physical layer and media access control for low-rate wireless personal area networks (LR-WPANs)): The wireless network communications uses an integrated radio transceiver, the TI CC2420 (formerly ChipCon). The CC2420 is IEEE 802.15.4 compliant and operates in the 2.4GHz to 2.4835GHz ISM unlicensed bands.
- battery connector, and
- daughterboard connector

3. PROBLEM STATEMENT

Sensor networks are vulnerable to resource consumption attacks. Computing power, memory, and battery life of devices are more constrained in WSN. Over the years many efforts are done to secure wireless sensor networks using less resources and many successful implementation has also been devised. Main objective is to explore the implementation of ECC on sunspot and compare it with RSA and see which one is better.

Objectives:

1. To study and explore various algorithms for transmission of data over WSN securely.
2. To implement RSA and ECC for WSN devices.
3. Demonstrate ECC and RSA on Sun Spot and compare the results.

4. IMPLEMENTATION AND RESULTS

1.3 Software used in experiment

All the different types of software used during experiment are discussed here

4.1.1 Sun Spot Manager

The Sun SPOT Manager tool is a Java WebStart Application for installing and managing Sun SPOT Software Development Kit (SDK) [28]. Minimum requirement to run Sun Spot Manager is at least a Sun Java Runtime Environment (JRE) is installed. After that Manager will itself install JDK, ANT and Netbeans is they are not installed on computer. Although sunspot manager will install JDK and ANT but if environment variables are not set accordingly every time it run it will ask to install JDK and ANT. Set environment variables as described in following sections.

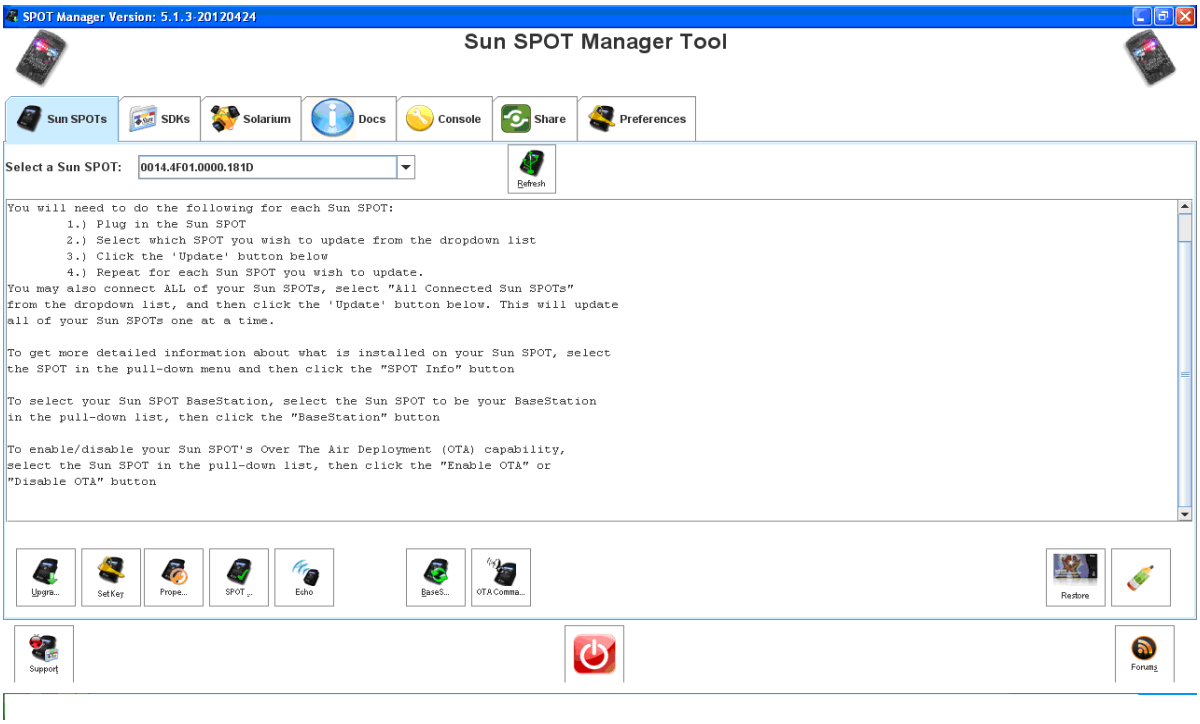


Figure 4.1 Sun Spot Manager Interface

4.1.2 Java Development Kit (JDK)

The JDK is an Oracle Corporation product aimed at Java developers. The JDK has collection of programming tools like javac, java, apt etc. jdk version used in implementation is jdk1.7.0_03. Set environment variable as JAVA_HOME= %path-of-jdk% and in Path variable add new path as ;%JAVA_HOME%\bin.

4.1.3 ANT

Apache Ant is a Java library and command-line tool. The main known usage of Ant is the build of Java applications. Ant supplies a number of built-in tasks allowing to compile, assemble, test and run Java applications [29]. The version used is apache-ant-1.8.2. Set environment variable as ANT_HOME=%path-of-ant% and in Path variable add new path as ;%ANT_HOME%\bin.

4.1.4 Netbeans

Netbeans is a free, open-source Integrated Development Environment for software developers. All the tools needed to create professional desktop, enterprise, web, and mobile applications with the Java platform, as well as with C/C++, PHP, JavaScript [30]. Since Sun Spot is java programmable, so developed code in netbeans and then deployed to Sun Spot. Select the project in the project window, right-click on it and select "Deploy to Sun Spot" from the pop-up menu. This will create a jar file from the compiled code and start to transfer it to the Sun SPOT. Version used is NetBeans 7.1.1.

4.1.5 Keytool

Java keytool is a key and certificate management utility. It allows users to manage their own public/private key pairs and certificates. Java keytool stores the keys and certificates in what is called a keystore. A keytool keystore contains the private key and any certificates necessary to complete a chain of trust and establish the trustworthiness of the primary certificate [31]. Keytool is provided in JDK.

4.1.6 OpenSSL

OpenSSL is an open-source implementation of the SSL and TLS protocols. The core library, written in the C programming language, implements the basic cryptographic functions and provides various utility functions [32].

4.2 Comparison of RSA and ECC

RSA most commonly used public-key cryptosystem. As due to advancement and increase in computing power available to an adversary, both symmetric and public key sizes must grow over time so that acceptable security for a fixed protection life span can be provided and Table 4.1 shows this expected key-size growth for various symmetric and public-key cryptosystems.

Table 4.1 Computationally equivalent key sizes [23]

Symmetric	ECC	RSA/DSA
80	163	1024
128	283	3072
192	409	7680
256	571	15360

ECC offers the highest strength per bit of any known public-key cryptosystem and as security demand is increasing and key size are growing to meet security demand, ECC is getting more attraction. ECC is better for constrained environment like WSN where smaller keys result in power, bandwidth and computational saving. Figure 4.2 shows the comparison with the key size generated with different algorithms and time to break each algorithm.

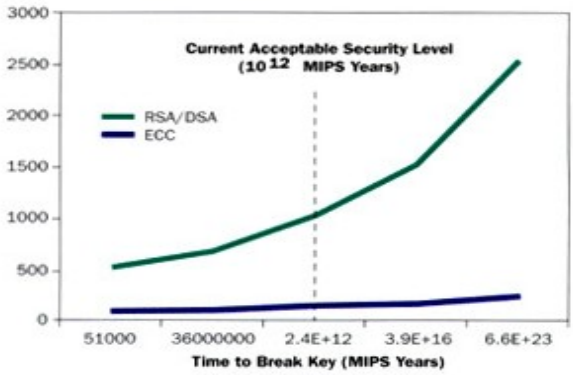


Figure 4.2 Comparison of Security level of RSA/DSA and ECC [33]

Energy Cost of RSA and ECC Operations

Table 4.2 compares the energy consumed by RSA and ECC for generating and verifying signatures. While the cost of an RSA verify is small but sign operation is very expensive, both of which are required for authentication. In comparison, ECDSA signatures are significantly cheaper than RSA signatures and ECDSA verifications are little expensive than RSA verification but its still in reasonable range. Important thing to note s that when transitioning is from RSA-1024 to RSA-2048 the energy cost of signing increases by a factor of more than seven, while ECDSA-224 signing is less than three times as expensive as ECDSA-160 signing.

Table 4.2 Energy Cost of digital signature computations [25]

Algorithm	Sign (mJ)	Verify (mJ)
RSA-1024	304	11.9
ECDSA-160	22.82	45.09
RSA-2048	2302.7	53.7
ECDSA-224	61.54	121.98

The RSA-based key exchange protocol heavily relies on A to encrypt a randomly generated secret key with B's public key, and B decrypting the key using its private key. With ECC, both parties perform a single ECDH operation to derive the secret

key. Table 4.3 compares the energy cost of key exchanges not including authentication and certificate verification. Energy calculated in millijoule (mJ). Difference lies of Server side where RSA consume lots of energy as compared to ECC.

Table 4.3 Energy cost of key exchange computations [25]

Algorithm	Client (mJ)	Server (mJ)
RSA-1024	15.4	304
ECDSA-160	22.3	22.3
RSA-2048	57.2	2302.7
ECDSA-224	60.4	60.4

Above RSA and ECC energy comparison is shown on different WSN devices. Here in this experiment comparison of ECC is done with RSA on Sun SPOT motes and energy consumption is analyzed.

2.3 Experiment Details

Here all steps used to create RSA and ECC certificates, deploy created certificate on SunSpot and Server so that data can be securely transferred and analyze the battery consumed by RSA and ECC is described.

4.3.1 **Create RSA/ECC certificate**

Here main discussion is about how to create RSA and ECC certificate using keytool and openssl which are added to trusted keys of Sunspot and Server so that secure data communication is achieved.

RSA Certificate

Keytool is used to create RSA certificate. Command is

" keytool -keystore mySrvKeyStore -genkeypair -keyalg RSA -keysize 1024"

If one is using JDK jdk1.6.0_25 or previous version the signature algorithm (-sigalg) by default is "SHA1with RSA" but in later default value of –sigalg is "SHA256withRSA". Since Sun Spot can only understand cipher suit of SHA1withRSA so –sigalg has to be specified with keytool using later version of jdk.

"keytool -keystore mySrvKeyStore -genkeypair -keyalg RSA -sigalg SHA1withRSA -keysize 1024"

Here new keystore named mySrvKeyStore is generated, -genkeypair generates public key and associated private key, algorithm used is RSA and algorithm keysize is 1024. As shown in Figure 4.3 When pressed enter its asks to enter keystore password then first name/last name (CN value which is your computer name or IP of any interface), name of organization etc. Since '-alias' is not mentioned it is set to default mykey. Figure 4.4 shows the mykey stored in keystore mySrvKeyStore.

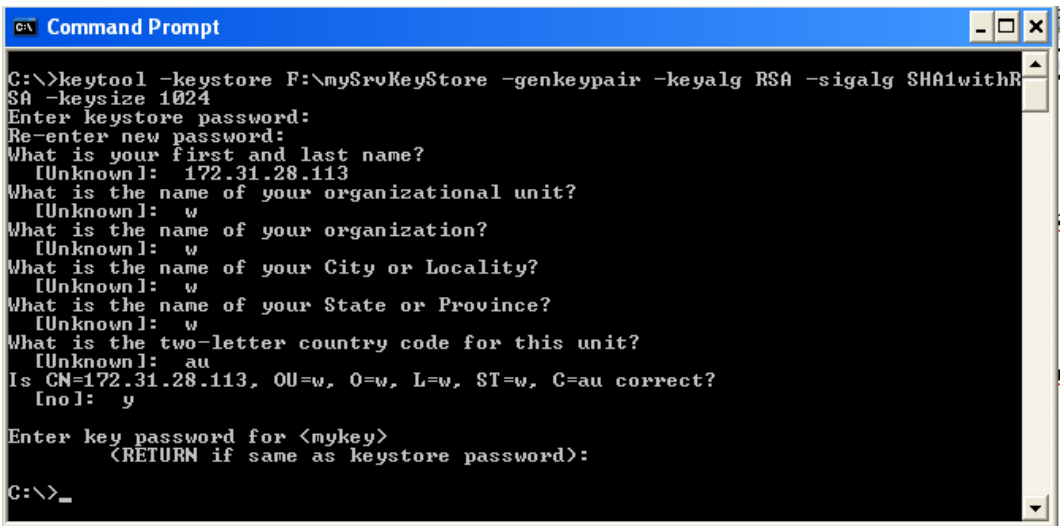


Figure 4.3 Keytool command to generate RSA key

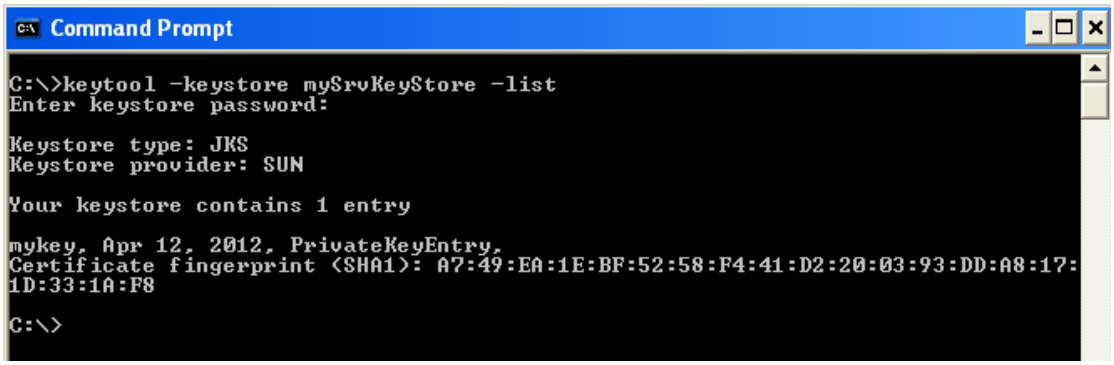


Figure 4.4 List keys in keystore

Exported the public key as a self-signed certificate into a file called mycert using "keytool -keystore mySrvKeyStore -exportcert -alias mykey -file mycert"

ECC Certificate

The Sun Spot library support only ECDH-ECDSA-RC4-SHA and secp160r1. OpenSSL is used to create to generate keys and certificate for curve secp160r1. Added %openSSL path%\bin to path in environment variables. For eg. C:\openssl\bin. Below are steps to create EC certificate and commands used to achieve it.

1. Generated self-signed CA certificate (on curve secp160r1). Command is-

   "openssl ecparam -name secp160r1 -out secp160r1.pem"

   '-name' specify the curve name (here it is secp160r1) '-out' output file name.

2. Generated a new certificate request.

   "openssl req C:/openssl/share/openssl.cnf -nodes -subj "/C=US/ST=CA/L=Mountain View/O=Sun Microsystems, Inc./OU=Sun Microsystems Laboratories/CN=172.31.28.113" -keyout secp160r1TestCA.key.pem -newkey ec:secp160r1.pem -new -out secp160r1TestCA.req.pem".

   A new certificate request is generated in secp160r1TestCA.req.pem. A new ECDSA (actually ECC) key pair is generated on the parameters in secp160r1.pem and the private key is saved in secp160r1TestCA.key.pem.

The ‘req’ command primarily creates and processes certificate requests in PKCS#10 format.

3. Signed the certificate request

   “openssl x509 -req -days 1500 -in secp160r1TestCA.req.pem -extfile C:openssl/share/openssl.cnf -extensions v3_ca -signkey secp160r1TestCA.key.pem -out secp160r1TestCA.cert.pem”.

   Signed the certificate request in secp160r1TestCA.req.pem using the private key in secp160r1TestCA.key.pem and included the CA extension. Made the certificate valid for 1500 days from the time of signing. The certificate is written into secp160r1TestCA.cert.pem. Figure 4.5 shows certificate created.

4. Place the certificate and key in a common file.

   “openssl x509 -in secp160r1TestCA.cert.pem -issuer –subject > secp160r1TestCA.pem”.

   “echo secp160r1TestCA.key.pem >> secp160r1TestCA.pem”.

5. Created Certificate secp160r1TestCA.pem cannot be added to trusted keys of Sun Spot so converted it into .der format.

   “openssl x509 -outform der -in secp160r1TestCA.pem -out secp160r1TestCA.der”

6. For Server which is running on computer. Change certificate created into an intermediate stage by exporting the key and certificate to a PKCS12 file. Import key and certificate into keystore using keytool command as explained below.

   “openssl pkcs12 -export -in secp160r1TestCA.cert.pem –inkey secp160r1TestCA.key.pem -out bundle.p12”.

   “keytool -importkeystore -deststorepass [new_keystore_pass] -destkeystore keystore.jks -srckeystore bundle.p12 -srcstoretype PKCS12 -srcstorepass [pass_used_in_p12_keystore] “

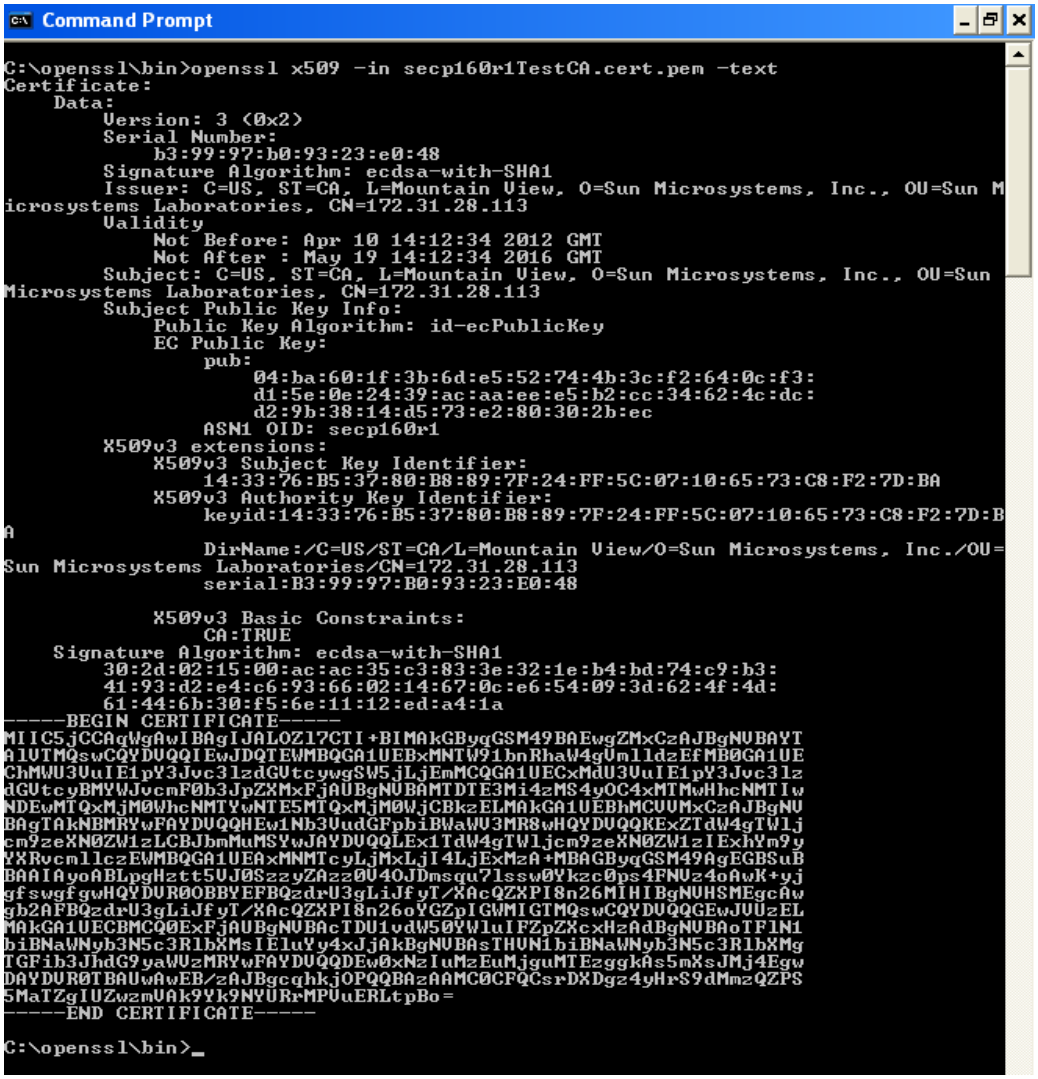


Figure 4.5 ECC certificate

Two keystore are created one for RSA and other for EC. mysrvKeystore has signature algorithm SHA1withRSA and keystore.jks has signature algorithm SHA1withECDSA as shown in Figure 4.6. These keystore created are used while running Server and mykey and secp160r1TestCA.der are added into trusted keys of Sunspot so that successful secure communication can be done between the two.

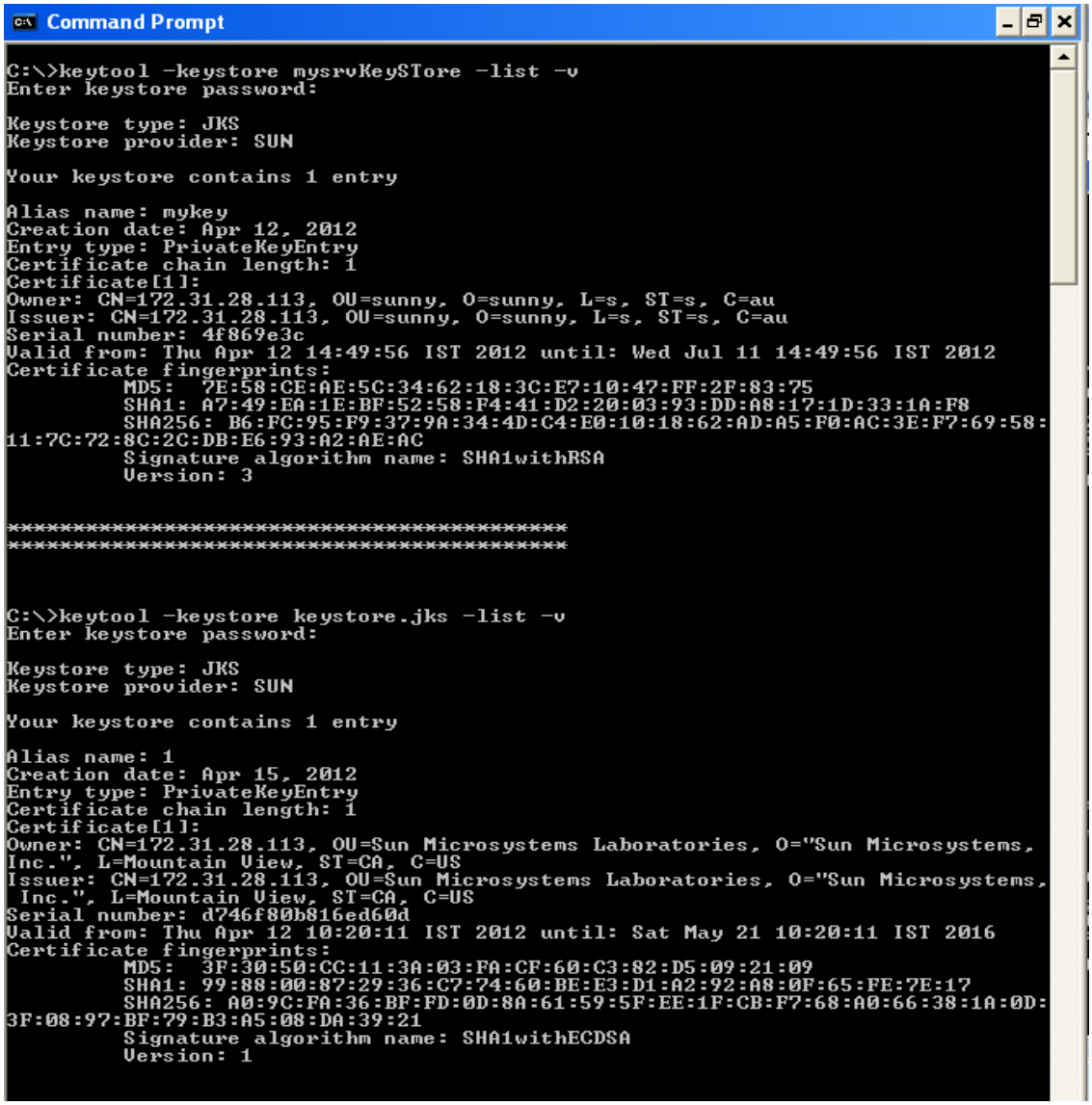


Figure 4.6 Keystore description

4.3.2 Experiment Setup and Explanation

The entire implementation is done using Java Sun Spot kit. Java program (Client) running on Sunspot send data using secure connection via base station to server running on computer. As explained, Sunspot only understand 3 cipher suits - TLS_ECDH_ECDSA_WITH_RC4_128_SHA, SSL_RSA_WITH_RC4_128_SHA, SSL_RSA_WITH_RC4_128_MD5. Figure 4.7 shows source code of program running on Sunspot and Figure 4.8 shows source code of Server used during implementation.

```
public class SensorSampler extends MIDlet {
  private static final int HOST_PORT = 1000;
  private static final int SAMPLE_PERIOD = 27*1000;   // in milliseconds

  protected void startApp() throws MIDletStateChangeException {
    StreamConnection conn;
    DataOutputStream outSPOT=null;
    String ourAddress = System.getProperty("IEEE_ADDRESS");
    IScalarInput lightSensor =  EDemoBoard.getInstance().getLightSensor();
    IBattery battery = Spot.getInstance().getPowerController().getBattery();
    long now;
    int reading = 0;
    double batt;
    int i;
    ITriColorLED[] leds = EDemoBoard.getInstance().getLEDs();
    try {

      conn = (StreamConnection)Connector.open("sslsocket://172.31.28.113:" + HOST_PORT,
        Connector.READ_WRITE, true);
     outSPOT = new DataOutputStream(conn.openOutputStream());
    }
    catch (Exception e) {
      System.err.println("Caught " + e + "in connection initialization.");
      System.exit(1);
    }
    //System.out.println("Starting sensing action and then send it to Server ... \n");
    while(true) {
      try {
        batt= battery.getAvailableCapacity() ;
        now = System.currentTimeMillis();
        reading = lightSensor.getValue();
        for(i=0;i<=7;i++)
        {
        leds[i].setRGB(255, 255, 255);
        leds[i].setOn();
        }
         outSPOT.writeInt(reading);
         outSPOT.writeDouble(batt);
         outSPOT.writeLong(now);
         outSPOT.flush();
        for (i=0;i<=7;i++)
        leds[i].setOff();
        Utils.sleep(SAMPLE_PERIOD - (System.currentTimeMillis() - now));
      }
```

Figure 4.7 Java code running on Sun Spot.

Serverv10.java - Notepad

File Edit Format View Help

```
import java.io.InputStream;
import java.io.DataInputStream;
import java.io.BufferedInputStream;
import java.net.ServerSocket;
import java.net.Socket;
import javax.net.ServerSocketFactory;
import javax.net.ssl.SSLServerSocketFactory;
import javax.net.ssl.SSLServerSocket;

public class Serverv10 {
       port in which the server is listening for connections
       private final static int PORT = 1000;

      public static void main (String[] argv) throws Exception {
      SSLServerSocketFactory ssocketFactory = (SSLServerSocketFactory)SSLServerSocketFactory.getDefault();
      SSLServerSocket ssocket = (SSLServerSocket)ssocketFactory.createServerSocket(PORT);

      System.out.println("Server running ... \n");

      String[] enabledCipherSuites = { "SSL_RSA_WITH_RC4_128_SHA"};
      ssocket.setEnabledCipherSuites(enabledCipherSuites);

      System.out.println("Server waiting for client ... \n");

      Socket socket = ssocket.accept();
      DataInputStream input=new DataInputStream(socket.getInputStream());
      System.out.println("Server connected to a client ... \n");

      while(true) {
         DataInputStream input=new DataInputStream(new BufferedInputStream(socket.getInputStream(),131072));
         int vall = input.readInt();
         double val = input.readDouble();
         long tim = input.readLong();
         System.out.println("Light reading is " + vall + "  Battery Level is " + val + "  Time is " + tim);
      }
   }
}

//TLS_ECDH_ECDSA_WITH_RC4_128_SHA
//SSL_RSA_WITH_RC4_128_SHA
```

Depending on which cipher suit to use
value of enabledCipherSuits will change

Figure 4.8 Server Code

Sun Spot Side Implementation

Sun Spot device is connected to the system via USB cable. Free Sun Spot should have OTA(over the air) enabled and base station should have OTA disabled. This can be verified using “ant info” command and by commands “ant enableota” and “ant disableota” one can achieve desired task.

Firstly rebuild the SPOT library code to include SSL and the SPOT-side code implementing new crypto related management commands. Since new code to extend SpotClient (the host-side portion of management commands) also need to be installed to match. Steps followed to do this are discussed below.

1. Connected Sun SPOT via a USB cable and ran command "ant deletepublickey" to put it in a keyless state.
2. Next, modified .sunspot.properties file which is created in the user's home directory as part of the SDK installation process. Replaced the existing line in this file that defines spot.library.addin.jars with the lines as shown in Figure 4.9

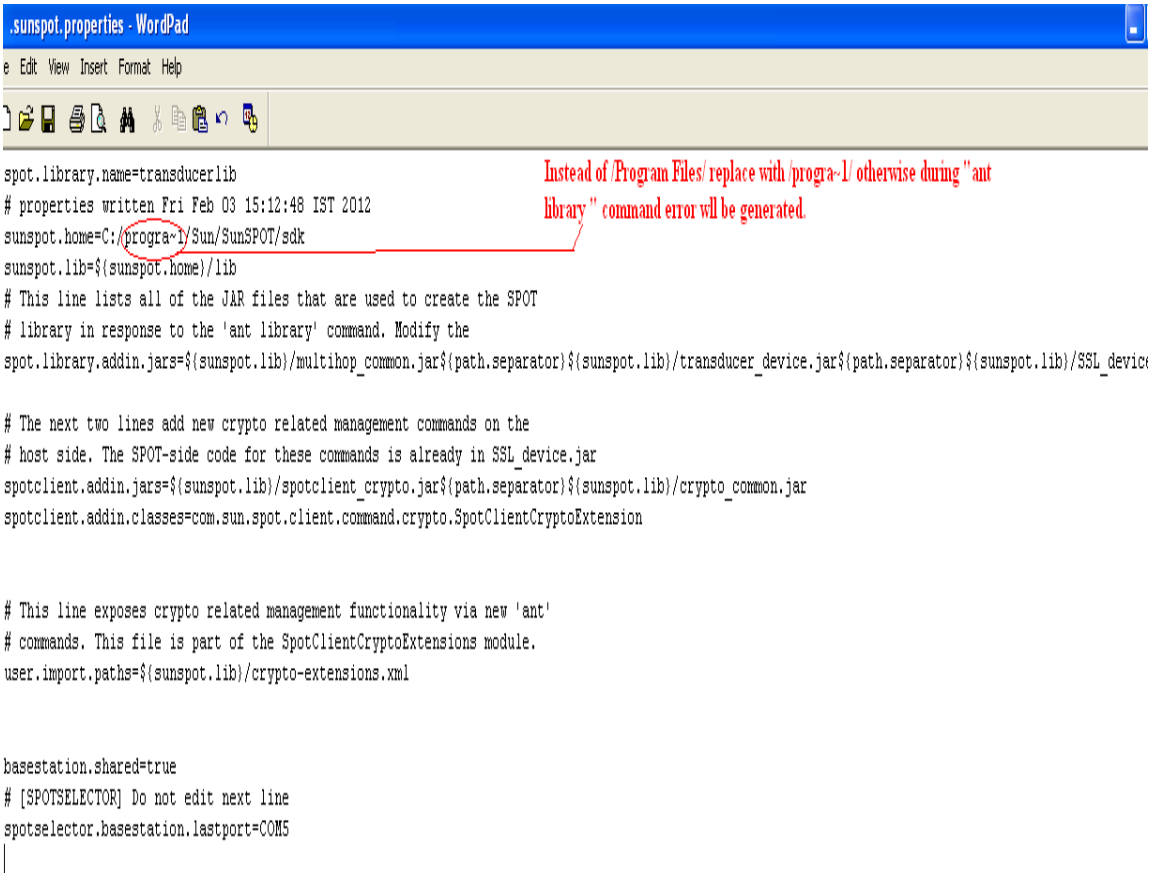


Figure 4.9 Sunspot Properties File

3. Build a new library including SSL_device.jar (this JAR already contains crypto_common.jar) and flash it on free-range SPOTs over a USB connection.

   " cd SDK_HOME " (here SDK_HOME refers to SDK installation directory)

   "ant library" (this only needs to be done once)

   "ant resetlibrary" (repeat this for each SPOT to be updated (connect SPOT via USB))

Clean and build sunspot java code already shown is Figure 4.7 in Netbeans and deployed on Sun Spot this is done via Netbeans or by ANT command "ant deploy" also this can achieved. Figure 4.10 shows code deployed through Netbeans.

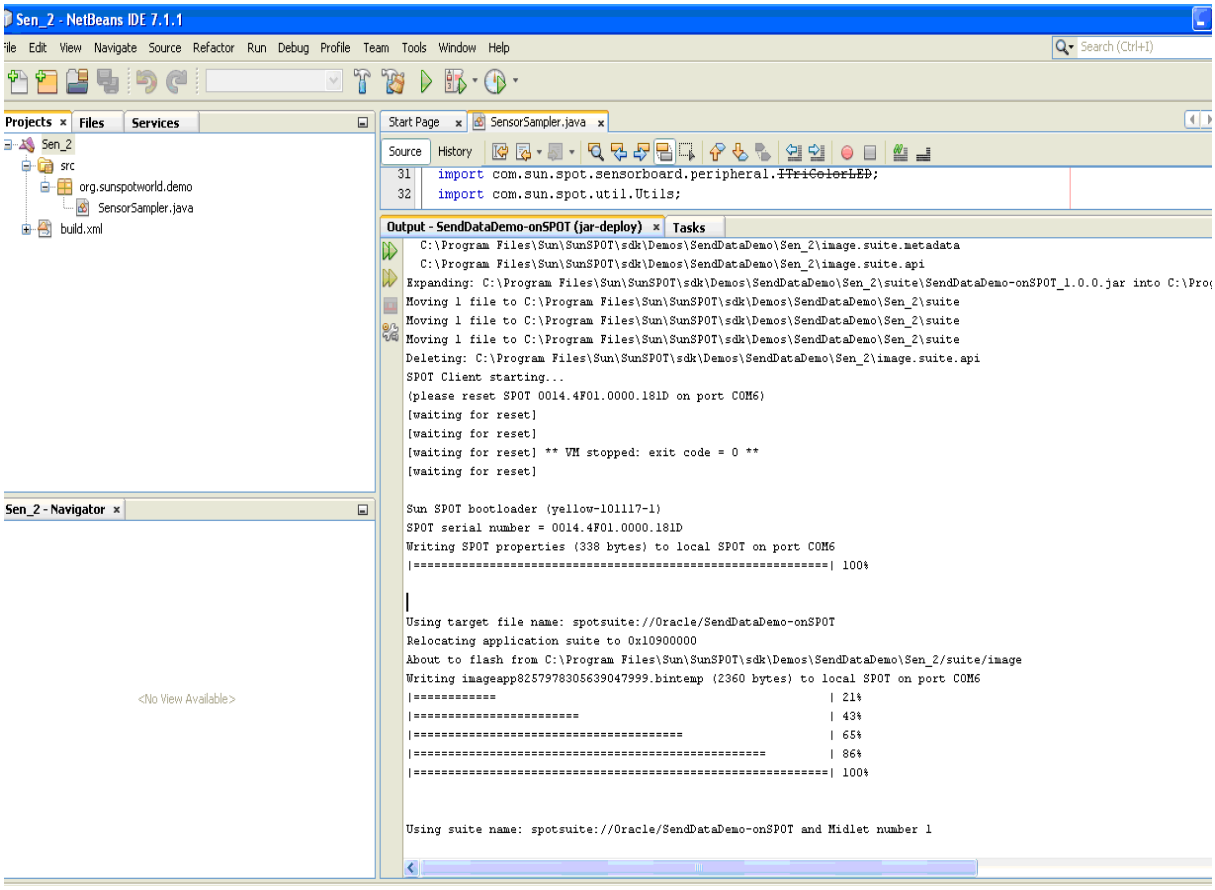


Figure 4.10 Code Deployed on Sun Spot

Next a Certificates are added which are created in Section 4.3.1 into trusted keys of Sun Spot, depending upon data is to be secured by which algorithm. First RSA file

“mycert” which has public key exported as self signed certificate is added, as already explained. Command is given below.

“ant addtrustedkey -Dcert=%path-of-mycert%\mycert –Dnickname=sunnyRSA –DtrustFlags={s/o/w}”

Everything is self explained, cert takes path of certificate, nickname is name with which user want to recognize this key, trustFlags value can be s, o, w where The flag 's' indicates self and 'o' indicates owner and ‘w’ flag is meant to indicate trust this key for signing certs used by secure Web servers running on the Internet.

Similarly, EC certificate secp160r1.der is added into trusted keys of Sun Spot.

“ant addtrustedkey -Dcert=%path-of-secp160r1.der%\secp160r1.der –Dnickname=sunnyEC –DtrustFlags={s/o/w}”

command “ant listtrustedkeys” lists all the trusted keys in keystore of sunspot. As shown in Figure 4.11. *MyCert and owner are by default in keystore of Sun Spot and sunnyRSA is added into its keystore.

```
     [java]    spot.battery.model: LP523436B
     [java]    spot.external.0.firmware.version: 1.14
     [java]    spot.external.0.hardware.rev: 5.0
     [java]    spot.external.0.part.id: EDEMOBOARD_REV_0_2_0_0
     [java]    spot.hardware.rev: 5
     [java]    spot.ota.enable: true
     [java]    spot.powercontroller.firmware.version: PCTRL-1.105
     [java]    spot.sdk.version: yellow-101117-1
     [java]    spot.startup.isolates.uriids: spotsuite://Oracle/SendDataDemo-onS
POT^1
     [java]
     [java] Keystore:
     [java]    Nickname      Subject                 Issuer              Flags
     [java]    *MyCert       CN=0014.4F01.0000.181D  CN=SDK-04c864c4     s
     [java]    owner         CN=SDK-04c864c4         CN=SDK-04c864c4     o
     [java]    sunnyRSA      C=au;ST=s;L=s;O=sunny;OU=sunny;CN=172.31.28.113C=au
;ST=s;L=s;O=sunny;OU=sunny;CN=172.31.28.113w
     [java]
     [java]
     [java] Exiting

-run-spotclient-multiple-times-locally:

-run-spotclient:

BUILD SUCCESSFUL
Total time: 10 seconds
```

Figure 4.11 Output of Ant listtrustedkeys command

To Delete key from keystore of Sun Spot command is:

“ant -Dnickname=<enter the name> deletetrustedkey”

To get details of specific key, command is:

“ant listtrustedkey –Dnickname=<enter nickname of that key>”

Figure 4.12 shows details of ECC key added to keystore of SunSPot and Fig 4.13 shows details of RSA

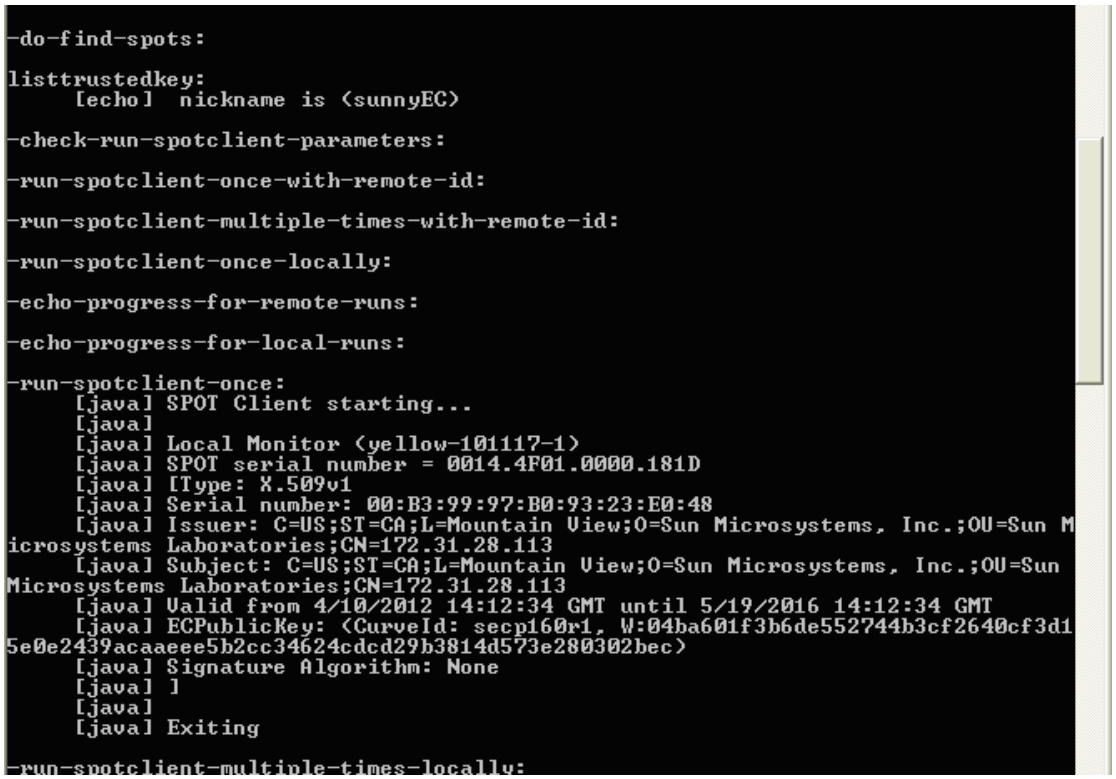


Figure 4.12 Details of ECC key in Sun Spot

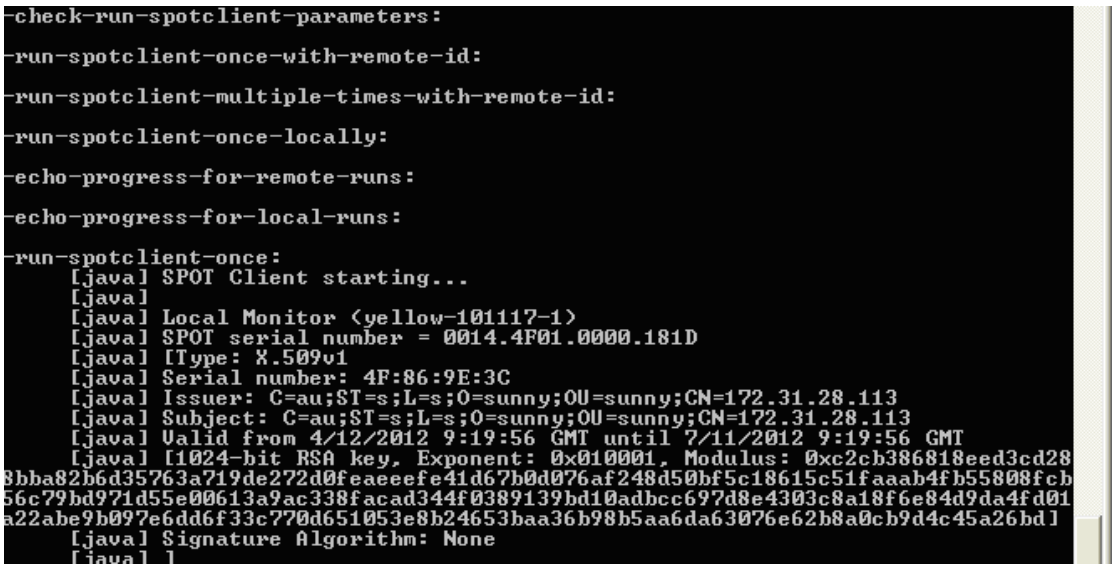


Figure 4.13 Details of RSA key in Sun Spot

Server Side Implementation

By default truststore is in "jre/lib/security/cacerts" but at runtime also keystore to be used by server can be given. Ran the Server code on terminal window using java command.

"Java -Djavax.net.ssl.keyStore=keystore -Djavax.net.ssl.keyStorePassword=paswrd Serverv10 >> ECtime.txt"

Here the output was not displayed on command prompt rather data was collected in txt file and keystore used was ECC keystore which was created 'keytsore.jks' and supplied password which was given at runtime.

Similarly, Server can be ran for RSA keystore created.

"Java -Djavax.net.ssl.keyStore=mysrvKeyStore -Djavax.net.ssl.keyStorePassword=paswrd Serverv10 >> RSAtime.txt"

To get details of all communication, ran the server with additional parameter '–Djavax.net.debug=all'. Figure 4.14 show sample output using the above said parameter while running server with cipher suit SSL_RSA_WITH_RC4_128_SHA and Figure 4.15 with TLS_ECDH_ECDSA_WITH_RC4_128_SHA

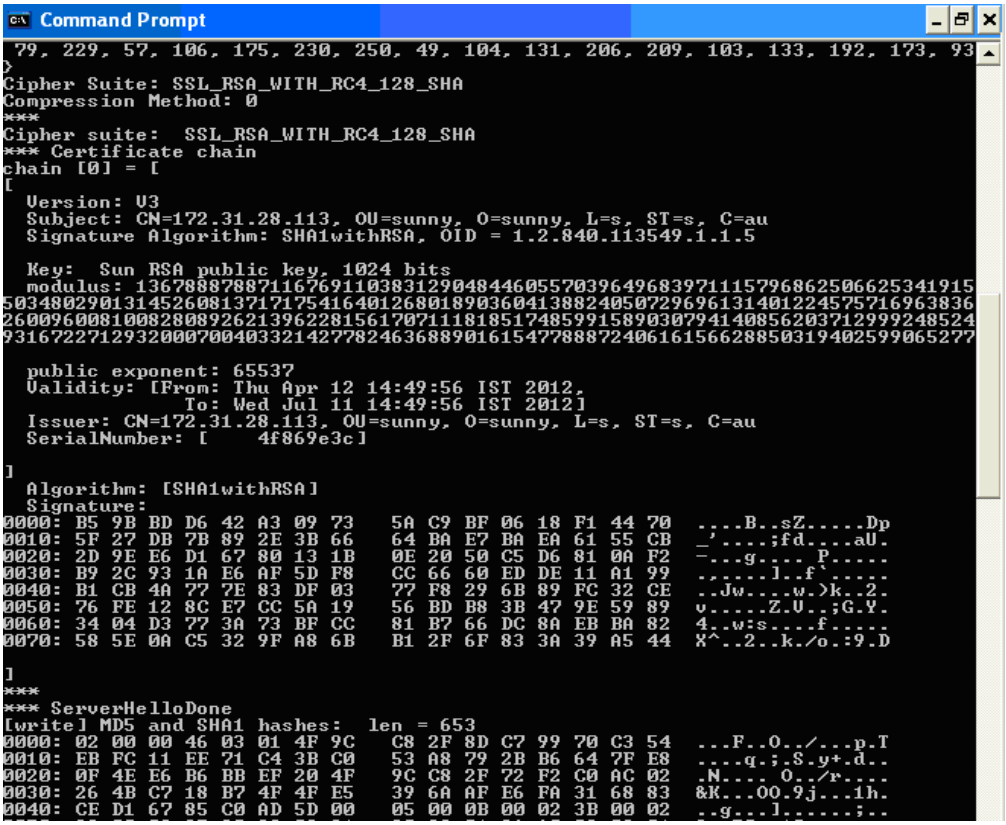


Figure 4.14 Sample of RSA handshake

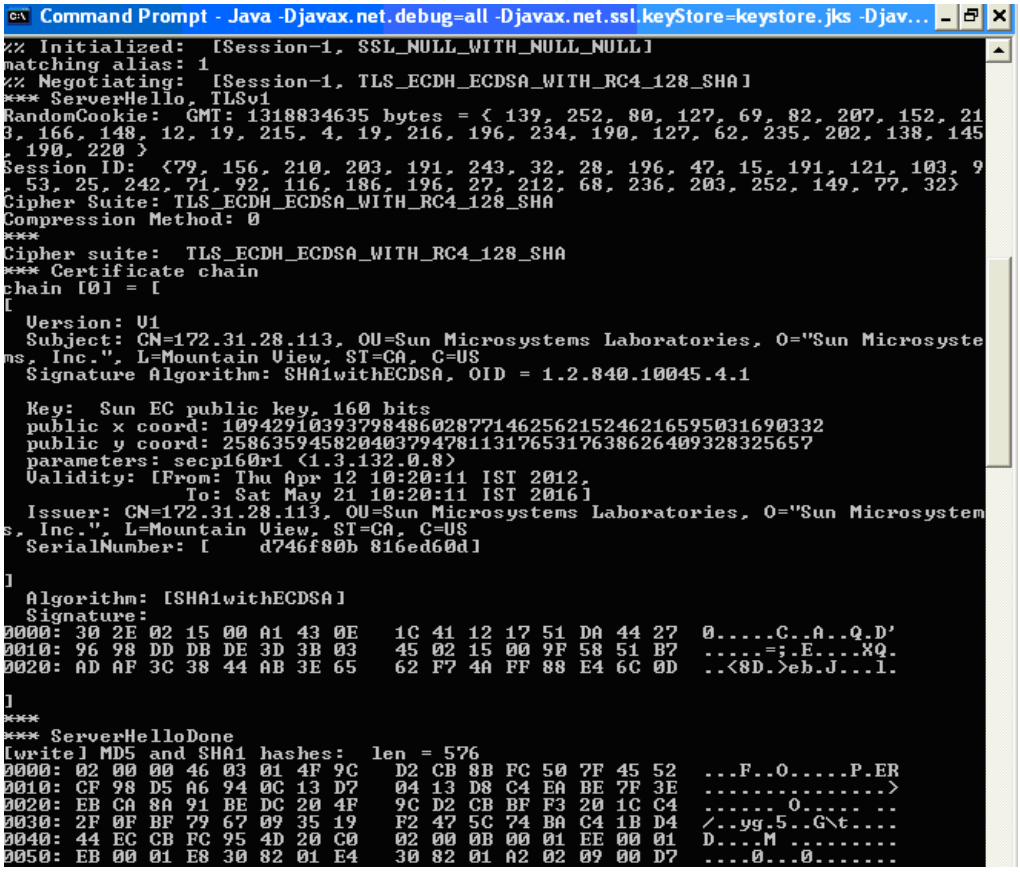


Figure 4.15 Sample of ECC handshake

Proxy Server

A proxy server is a server that acts as an intermediary for requests from clients seeking resources from other servers.

Started the socket proxy on a host machine in a different terminal window. Command line proxy is run by command "ant socket-proxy" and gui based proxy is run by command "ant socket-proxy-gui ". In this experiment gui based proxy was used.

Then clicked the 'Start' button. Figure 4.16 shows output of ant socket-proxy-gui command.

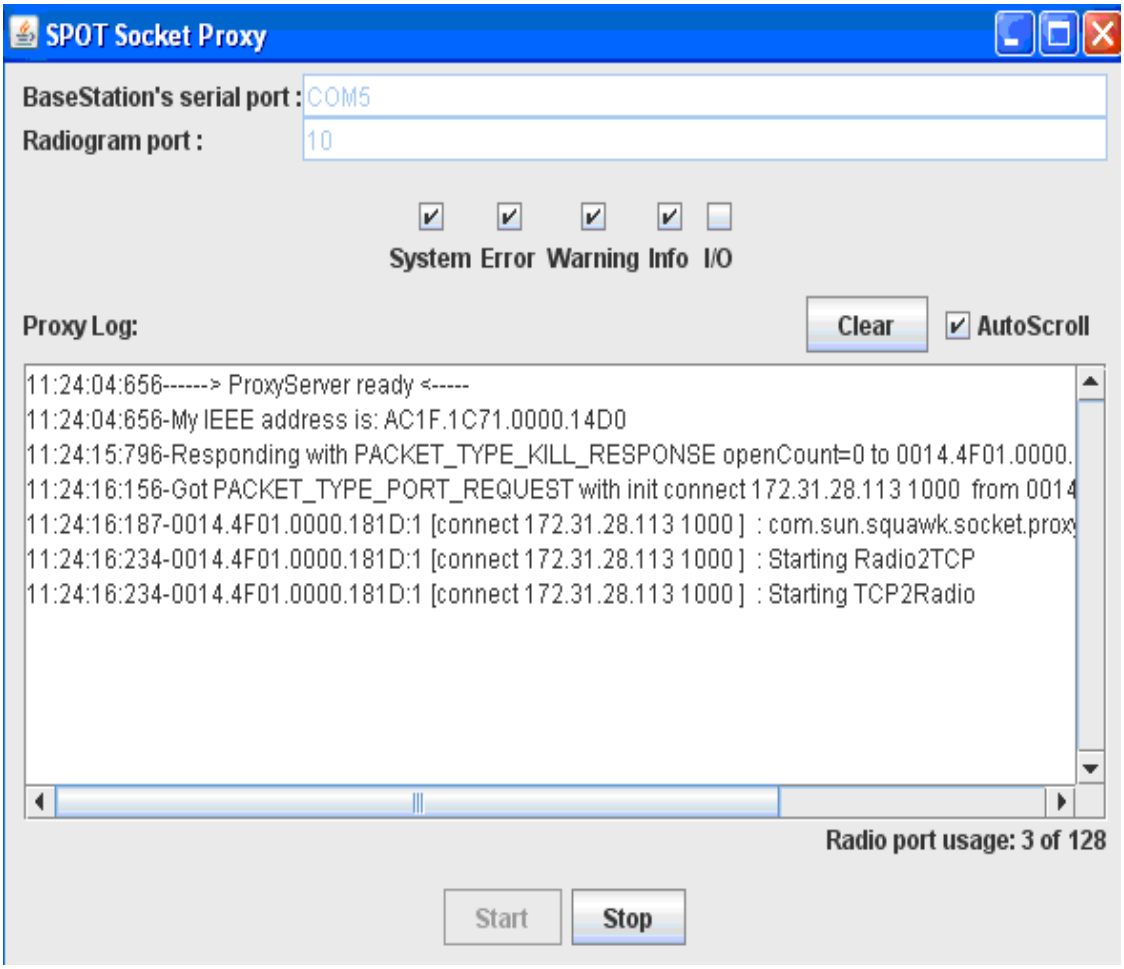


Figure 4.16 Output of ant socket-proxy-gui.

4.4 Results

As per the literature Sun Spot takes around 3-4 hours to charge it fully via USB. Here in this experiment Sun Spot is charged for 4 hours and then run once with RSA algorithm and next time with ECC algorithm and find that battery sustain longer time when data is encrypted using ECC algorithm than RSA algorithm. As already mentioned data collected is light reading, battery value and time (in milliseconds). The time mentioned is in milliseconds which is calculated as milliseconds from 1 January 1970, 00:00 GMT till that time. Few readings of output is shown in Figure 4.17 when data is encrypted using RSA algorithm and in Figure 4.18 when data is encrypted using ECC algorithm.

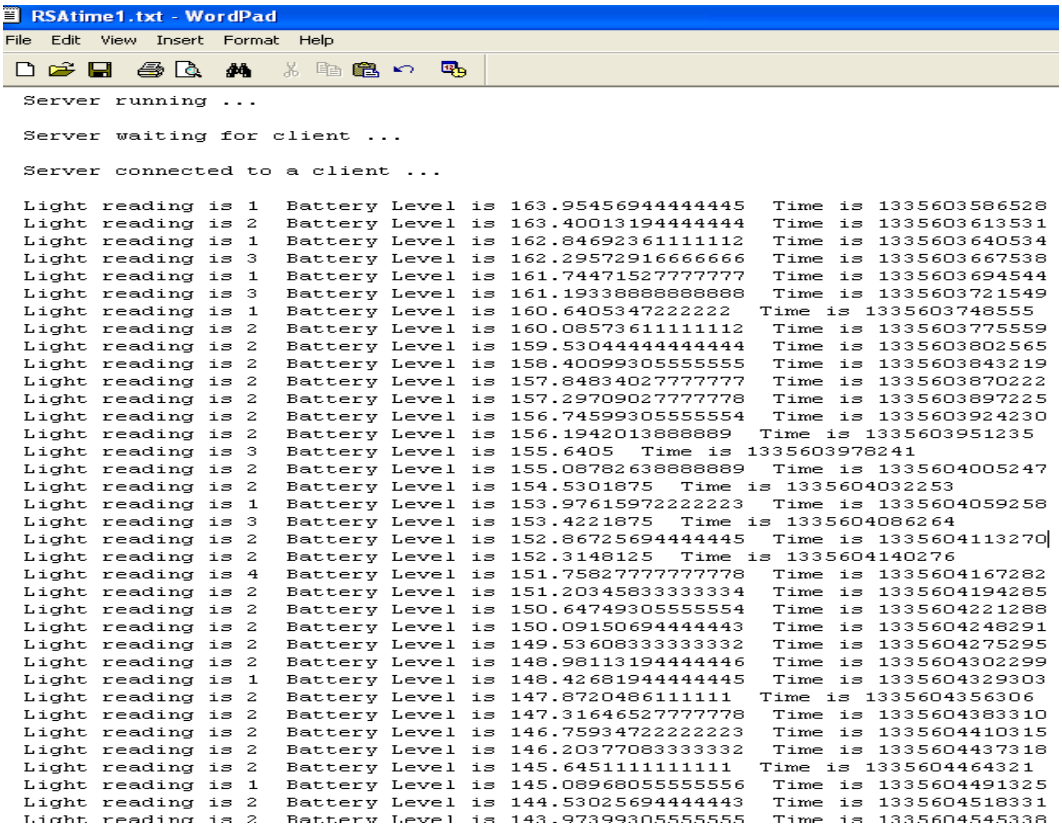


Figure 4.17 Output using RSA algorithm

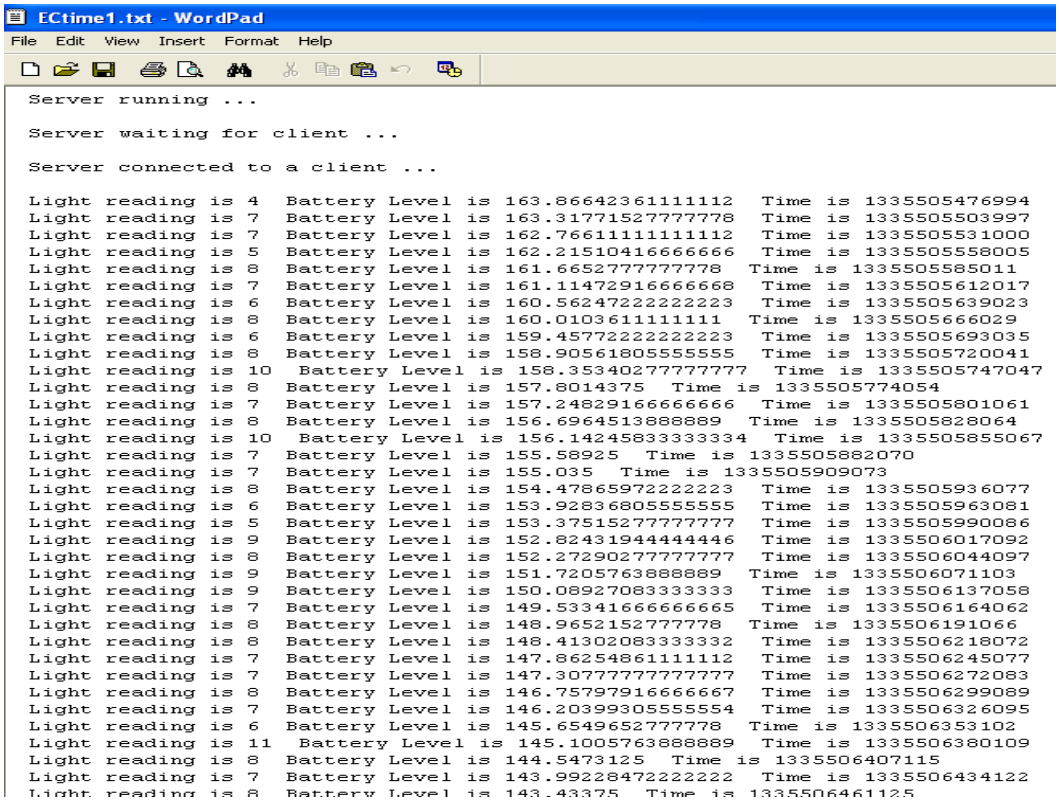


Figure 4.18 Output using ECC algorithm

To calculate the total time Sun Spot take to discharge whole battery, difference of last reading and first reading is taken. Table 4.4 shows first and last reading using RSA algorithm and Table 4.5 shows first and last reading using EC algorithm.

Table 4.4 Output recorded using RSA

Reading No	Light Reading	Available Battery	Time (in milliseconds)
First	1	163.95456944444445	1335603586528
Last	2	16.0	1335610487738

Table 4.5 Output recorded using ECC

Reading No	Light Reading	Available Battery	Time (in milliseconds)
First	4	163.86642361111112	1335505476994
Last	9	16.0	1335512917294

Under same circumstances, from above readings it is observed that by using RSA algorithm for encryption battery discharges in 6901210 milliseconds (1335610487738-1335603586528) i.e approx. 115.020 minutes or 1.917 hrs. By using ECC algorithm for encryption battery discharges in 7440300 milliseconds (1335505476994- 1335505476994) i.e. approx. 124.005 minutes or 2.066. Experiment is ran 10 times for each algorithm and it is observed that battery consumption is less when ECC algorithm is used to encrypt data as compared to RSA.

It was observed that battery level never reaches zero rather it never drops beyond reading 16. In Sun Spot the battery level is computed based on the measured discharge rate and so is approximate. It is also computed conservatively which is why it goes to 16 after about few hrs and stays there until the battery voltage really falls at the end [8].

Its to be kept in mind that Sun Spot will run for longer duration if one increase its sleep time. Actually for most percentage of time Sun Spot is sleeping and only for few milliseconds of time it actually operates. Changing lights of Sun Spot also saves energy. Green light takes least energy. Sun Spot can run for 7-12 hours depending upon its sleep time, Led lights used. But under same circumstances when encoded data is transferred ECC algorithm takes less battery energy than RSA algorithm as experimentally proved above.

5. CONCLUSION

This thesis investigated the mathematical foundation of RSA and elliptic curve cryptography for the purpose of understanding the practical problems of implementation of RSA and ECC over wireless sensor network.

Designed and implemented RSA and ECC for establishing secure connection in wireless sensor networks. At lower key bit size ECC provide same level of security as RSA.

Implemented the algorithms one by one over Sun SPOTs for analyzing the cryptographic behavior. Here Sun SPOT send the light reading , available battery capacity and time.

First charged the battery and then calculated the time taken by each algorithm to discharge the battery to find out which algorithm takes less amount of battery power or consumes less battery. Under same circumstances RSA consumes more battery power as compared to EC.

REFERENCES

[1] Matt Welsh, Dan Myung, Mark Gaynor, and Steve Moulton “Resuscitation monitoring with a wireless sensor network”, in Supplement to Circulation: Journal of the American Heart Association, October 2003.

[2] Jason Lester Hill “System Architecture for Wireless Sensor Networks”, University of California, Berkeley, Spring, 2003.

[3] Adrian Perrig, John Stankovic, David Wagner, “Security in Wireless Sensor Networks”, in Communications of The ACM, Vol 47, No.6, June 2004.

[4] Mayank Saraogi, “Security in Wireless Sensor Networks”, Department of Computer Science University of Tennessee, Knoxville, 2005.

[5] Y.C. Hu, A. Perrig, and D. B. Johnson, “Wormhole detection in wireless ad hoc networks,” Department of Computer Science, Rice University, Tech. Rep. TR01-384, June 2002.

[6] Ritesh Maheshwari, Jie Gao and Samir R. Das, “Detecting Wormhole Attacks in Wireless Networks using Connectivity Information”, Proc. 26th IEEE International Conference on Computer Communications (IEEE INFOCOM 2007 Conference), Anchorage, Alaska, May 2007.

[7] C. Karlof and D. Wagner, “Secure routing in wireless sensor networks: Attacks and countermeasures”, Elsevier's Ad Hoc Network Journal, Special Issue on Sensor Network Applications and Protocols, pp. 293-315, September 2003.

[8] Siwaruk Siwamogsatham, Songrit Srilasak, Kitiwat Limmongkol, and Kitti Wongthavaraw, “Encryption vs. Performance of Infrastructure IEEE 802.11 WLANs”, National Electronics and Computer Technology Center, Thailand, 2008.

[9] Technology Briefing Cryptography: A Security Tool of the Information Age Sergai Boukhonine available at www.bauer.uh.edu/uhisrc/FTB/Cryptography/FTBCryptography.pdf.

[10] Anoop MS, “Public Key Cryptography - Applications Algorithms and Mathematical Explanations”, Tata Elxsi Ltd, India, May 2007.

[11] H. Wang, B. Sheng, and Q. Li, "Elliptic curve cryptography-based access control in sensor networks," International Journal of Security and Networks, vol. 1, pp. 127-137, 2006.

[12] W. Diffie & M. Hellman, “New directions in cryptography”, IEEE Transaction on Information Theory, IT-22(6), pp. 644-654, 1976.

[13] RSA Laboratories, http://www.rsa.com/rsalabs/node.asp?id=2157. Accessed April 2, 2012.

[14] Anoop MS, “Security needs in embedded systems”, Tata Elxsi Ltd, India, May 2008.

[15] Amanda Back , “The Diffie-Hellman Key Exchange”, December 2,2009, available at http://129.81.170.14/~erowland/courses/2009-2/projects/Back.pdf.

[16] http://imps.mcmaster.ca/courses/SE-4C03-07/wiki/wrighd/rsa_alg.html. Accessed May 14, 2012.

[17] http://www.enotes.com/topic/Digital_Signature_Algorithm. Accessed May 14, 2012.

[18] V. Miller, "Use of elliptic curves in cryptography, Advances in Cryptology − CRYPTO 85", Springer-Verlag, pp. 417-426, 1985.

[19] N. Koblitz, "Elliptic curve cryptosystems", Mathematics of Computation, pp 203-209, 1987.

[20] David Greenberg, "Elliptic Curve Cryptography", May 14, 2009 available at http://web.mit.edu/dgrnbrg/www/eccpaper.pdf.

[21] Arun kumar, Dr. S.S. Tyagi, Manisha Rana, Neha Aggarwal, Pawan Bhadana, "A Comparative Study of Public Key Cryptosystem based on ECC and RSA", vol. 3, no. 5, pp 1906-1907, May 2011.

[22] Anoop MS, "Elliptic Curve Cryptography- An implementation tutorial", Tata Elxsi Ltd, India.

[23] Vipul Gupta, Sumit Gupta, Sheueling Chang, "Performance Analysis of Elliptic Curve Cryptography for SSL" Performance analysis of elliptic curve cryptography for SSL. In *Proc. 3rd ACM Workshop on Wireless Security*, pp. 87–94. ACM Press, 2002.

[24] S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", May 2006 available at http://tools.ietf.org/pdf/rfc4492.pdf.

[25] Arvinderpal S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, Sheueling Chang Shantz, "Energy Analysis of Public-Key Cryptography on Small Wireless Devices", *IEEE PerCom 2005*, pp. 324-328, Mar. 2005.

[26] Sun Labs, "Sun SPOT Programmer's Manual", copyright SunMicrosystems available at http://www.sunspotworld.com/docs/Yellow/SunSPOT-Programmers-Manual.pdf. Accessed April 4, 2012.

[27] Sun Labs, "Sun SPOT Theory of Operation", copyright SunMicrosystems available at http://sunspotworld.com/docs/Yellow/SunSPOT-TheoryOfOperation.pdf. Accessed March 28, 2012.

[28] Sun Labs, "http://www.sunspotworld.com/SPOTManager". Accessed May 8, 2012.

[29] Apache Ant, "http://ant.apache.org". Accessed May 7, 2012.

[30] Netbeans, "http://netbeans.org/features/index.html". Accessed May 2, 2012.

[31] Oracle, http://docs.oracle.com/javase/1.3/docs/tooldocs/win32/keytool.html. Accessed April 30, 2012.

[32] OpenSSL, "http://www.openssl.org/". Accessed April 30, 2012.

[33] http://www.opengroup.org/comm/the_message/magazine/mmv4n5/approach.htm. Accessed May 4, 2012.

List of Publications

"Energy Analysis of WSN using RSA and ECC Encryption method", Sharanjit Kaur, Maninder Singh, "International Journal of Computer Applications", communicated, June 2012.