



p- 3513



**A DYNAMIC CRYPTOGRAPHIC ALGORITHM FOR WIRELESS
SENSOR NETWORK**

By

M.MUTHUKUMAR

Reg. No. 0920107010

of

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology, Coimbatore)

COIMBATORE – 641 049

A PROJECT REPORT

Submitted to the

**FACULTY OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

In partial fulfillment of the requirements

for the award of the degree

of

MASTER OF ENGINEERING

IN

COMMUNICATION SYSTEMS

APRIL 2011

BONAFIDE CERTIFICATE

Certified that this project report entitled “A DYNAMIC CRYPTOGRAPHIC ALGORITHM FOR WIRELESS SENSOR NETWORK” is the bonafide work of Mr.M.Muthukumar [Reg. no. 0920107010] who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.



Project Guide

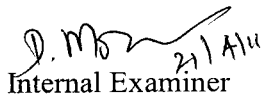
Mrs.M.Alagumeenaakshi



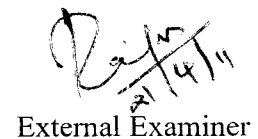
Head of the Department

Dr.Rajeswari Mariappan

The candidate with university Register no. 0920107010 is examined by us in the project viva-voce examination held on ..21.4.2011.....



Internal Examiner



External Examiner

ACKNOWLEDGEMENT

First I would like to express my praise and gratitude to the Lord, who has showered his grace and blessing enabling me to complete this project in an excellent manner.

I express my sincere thanks to our beloved Director **Dr.J.Shanmugam**, Kumaraguru College of Technology, for the kind support and necessary facilities to carry out the project work.

I express my sincere thanks to our beloved Principal **Dr.S.Ramachandran**, Kumaraguru College of Technology, who encouraged and supported me the project work.

I would like to express my sincere thanks and deep sense of gratitude to our HOD, **Dr.Rajeswari Mariappan,Ph.D** Department of Electronics and Communication Engineering, for the valuable suggestions and encouragement which paved way for the successful completion of the project work.

In particular, I wish to thank and everlasting gratitude to the project coordinator **D.Mohanageetha(Ph.D)**., Associate Professor, Department of Electronics and Communication Engineering for her expert counseling and guidance to make this project to a great deal of success.

I am greatly privileged to express deep sense of gratitude to my guide **Mrs.M.Alagumeenaakshi M.E.**, Assistant Professor (SRG), Department of Electronics and Communication Engineering, for the valuable suggestions and support throughout the course of this project work. I wish to convey my deep sense of gratitude to all the teaching and non-teaching staff of ECE Department for their help and cooperation.

Finally, I thank my parents and my family members for giving me the moral support and abundant blessings in all of my activities and my dear friends who helped me to endure my difficult times with their unflinching support and warm wishes.

ABSTRACT

Wireless sensor networks have many applications, vary in size, and are deployed in a wide variety of areas. They are often deployed in potentially adverse or even hostile environment so that there are some concerns on security issues in these networks. Sensor nodes used to form these networks are resource-constrained, which make security applications a challenging problem. Many applications of wireless sensor networks that collect and disseminate sensitive and important information, so as to maintain the privacy and security of the data stored in the sensor nodes and of the transmitted data. In this project a unique, dynamic cryptographic algorithm is proposed to provide security to the wireless sensor network by securing the individual nodes of the network.

AES algorithm is used to provide security over the channel and also the Dynamic key (Generated by RNG) is used to provide nodal level security in wireless sensor network. Simulation of the proposed algorithm is implemented using Network Simulator (NS-2).

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	iv
	LIST OF FIGURES	viii
	LIST OF TABLES	x
	LIST OF ABBREVIATIONS	xi
1	INTRODUCTION	1
	1.1 Overview of wireless sensor networks	1
	1.1.1 Wireless sensor network	1
	1.1.2 Sensing process	2
	1.1.3 Components of sensor networks	3
	1.2 Sensor network Architecture	4
	1.3 Characteristics of wireless sensor network	6
	1.3.1 Sensor node	6
	1.4 Applications of sensor networks	6
	1.5 Data collection methodology	7
	1.5.1 Event based data collection	8
	1.5.2 Periodic data collection	8
	1.6 Challenges of wireless sensor networks	9
2	LITERATURE SURVEY	13
	2.1 Random key predistribution schemes for sensor Networks	13
	2.2 A key management scheme in distributed sensor Networks using attack probabilities	14
	2.3 Energy-efficient communication protocol for Wireless micro sensor networks	14

2.4	Single network-wide key	15
2.5	Security concerns with respect to this document	15
2.6	Potential threats and attack scenarios	15
3	ADVANCED ENCRYPTION STANDARD (AES)	16
3.1	Introduction	16
3.2	Definitions	16
3.2.1	Glossary of terms and Acronyms	15
3.2.2	Algorithm parameters, symbols and definitions	18
3.3	Notation and conventions	19
3.3.1	Inputs and outputs	19
3.3.2	Bytes	19
3.3.3	Array of bytes	20
3.3.4	The state	21
3.3.5	The state as an array of columns	22
3.4	Mathematical preliminaries	22
3.5	Algorithm specification	22
3.5.1	Cipher	23
3.5.1.1	SubBytes transformation	25
3.5.1.2	ShiftRows transformation	26
3.5.1.3	MixColumns transformation	27
3.5.1.4	AddRoundkey transformation	28
3.5.2	Key expansion	29
3.5.3	Inverse cipher	30
3.5.3.1	InvShiftRows transformation	30
3.5.3.2	InvSubBytes transformation	31
3.5.3.3	InvMixColumns transformation	31
3.5.3.4	Inverse of the AddRoundKey transformation	32
3.5.3.5	Equivalent inverse cipher	32
4	PROPOSED SCHEME	34
4.1	Demerits of AES	34

4.2	Flow chart	35
4.3	Concept	36
4.4	Advantages of Dynamic key	37
5	ROUTING PROTOCOL	38
5.1	DSDV	38
5.2	Analysis of AODV & DSDV	39
5.3	Delay	40
5.4	Packet Loss	40
5.5	Random number generation	41
5.6	Seeding the RNG	42
6	RESULTS AND DISCUSSION	43
6.1	The simulation on NS-2	43
6.2	Results and Analysis using NAM	43
6.3	Physical/Mac/Network layers specifications	44
6.4	Performance metrics	44
6.5	Performance results	46
6.6	Inference	50
7	CONCLUSION	51
8	FUTURE WORK	52
	REFERENCES	

LIST OF FIGURES

FIGURE NO	CAPTION	PAGE NO
1.1.2[a]	Sensing Process	2
1.1.2[b]	Sensor nodes scattered in sensor field	3
1.1.3[a]	Components of sensor networks	3
1.2	Network Models: Hierarchical and Distributed Wireless Sensor Networks	5
1.5[a]	Data Collection in WSN	7
3.3.2[a]	Hexadecimal representation of bit patterns	20
3.3.3[a]	Indices for Bytes and Bits	21
3.3.4[a]	State array input and output	21
3.5	Key-Block-Round Combinations	23
3.5.1[a]	AES parameters	24
3.5.1[b]	AES Encryption and Decryption	24
3.5.1.1[a]	SubBytes() applies the S-box to each byte of the State	25
3.5.1.1[b]	S-box: substitution values for the byte xy (in hexadecimal format)	26
3.5.1.2[a]	ShiftRows() cyclically shifts the last three rows in the State	27

3.5.1.3[a]	MixColumns() operates on the State column-by-column.	28
3.5.1.4[a]	AddRoundKey() XORs each column of the State with a word from the key schedule.	29
3.5.3.1[a]	InvShiftRows() cyclically shifts the last three rows in the State.	31
3.5.3.2[a]	Inverse S-box: substitution values for the byte xy (in Hexadecimal format)	31
4.2[a]	Flow chart	35
5.1[b]	AODV Delay	38
5.1[c]	DSDV Delay	39
5.1[d]	Comparison of Delay between AODV & DSDV Routing protocol.	39
5.4[a]	Packet loss of AODV	40
5.4[b]	Packet loss of DSDV	40
5.5[a]	Overall arrangement of streams and substreams. [LSCCK01]	41
6.5[a]	17 nodes simulation	46
6.5 [b]	30 nodes simulation	47
6.5[c]	51 nodes simulation	47
6.5[d]	Overhead	48
6.5[e]	Throughput	48
6.5[f]	Packet Delivery Ratio	49
6.5[g]	PDR	49
6.5[h]	Delay	50

LIST OF TABLES

TABLE NO	CAPTION	PAGE NO
5.1[a]	Average and Total Delay between AODV & DSDV	38
6.1	Simulation settings and parameters of trust base technique	44

LIST OF ABBREVIATIONS

DSDV	-----	Destination Sequence Distance Vector
WSN	-----	Wireless Sensor Network
TCP	-----	Traffic Control Protocol
UDP	-----	User Datagram Protocol
FTP	-----	File Transfer Protocol
PDR	-----	Packet Delivery Ratio
AODV	-----	Ad hoc On – Demand Distance Vector

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF WIRELESS SENSOR NETWORKS

1.1.1 Wireless Sensor Network

Sensors are inexpensive, low-power devices which have limited resources. They are small in size, and have wireless communication capability within short distances. A sensor node typically contains a power unit, a sensing unit, a processing unit, a storage unit, and a wireless transmitter / receiver. A wireless sensor network (WSN) is composed of large number of sensor nodes with limited power, computation, storage and communication capabilities.

Environments, where sensor nodes are deployed, can be controlled (such as home, office, warehouse, forest, etc.) or uncontrolled (such as hostile or disaster areas, toxic regions, etc.). If the environment is known and under control, deployment may be achieved manually to establish an infrastructure. However, manual deployments become infeasible or even impossible as the number of the nodes increases.

If the environment is uncontrolled or the WSN is very large, deployment has to be performed by randomly scattering the sensor nodes to target area. It may be possible to provide denser sensor deployment at certain spots, but exact positions of the sensor nodes cannot be controlled. Thus, network topology cannot be known precisely prior to deployment. Although topology information can be obtained by using mobile sensor nodes and self-deployment protocols, this may not be possible for a large scale WSN.

1.1.2 Sensing process

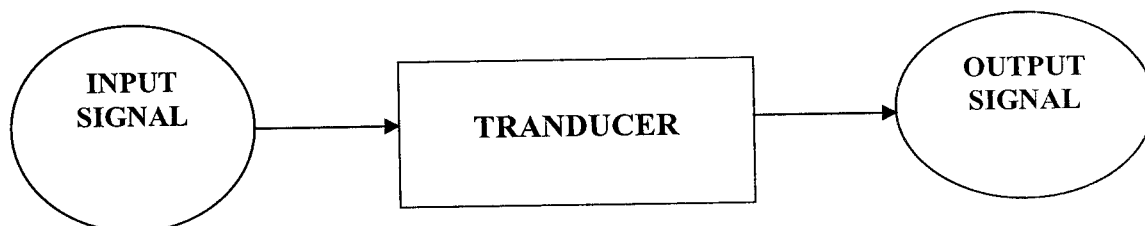


Figure 1.1.2[a] Sensing Process

Sensor networks, once deployed, are left unattended and expected to work for extended periods of time. Data is being sensed by the sensor nodes in the network must be transmitted to a processing center (base station / sink), where the end user can access the data. A sensor node can communicate directly only with other sensor nodes that are within a small distance. So all the sensor nodes cannot reach the processing center effectively, resulting in more energy consumption to transmit data than the order nodes and hence, exhausts and die sooner. Energy is a scarce resource for sensor systems and has to be managed wisely in order to prolong the life time of the sensor nodes for the duration of a particular mission.

The sensor nodes are usually scattered in a sensor field as shown in Fig 1.1.2[b]. Each of these scattered sensor nodes has the capabilities to collect data and route data back to the sink. Data are routed back to the sink by multi-hop infrastructure less architecture through the sink. The sink may communicate with the task manager node via Internet and satellite.

Sensor networks have wide application in areas such as environment and habitat monitoring, health care, military, collecting information in disaster-prone areas and surveillance application

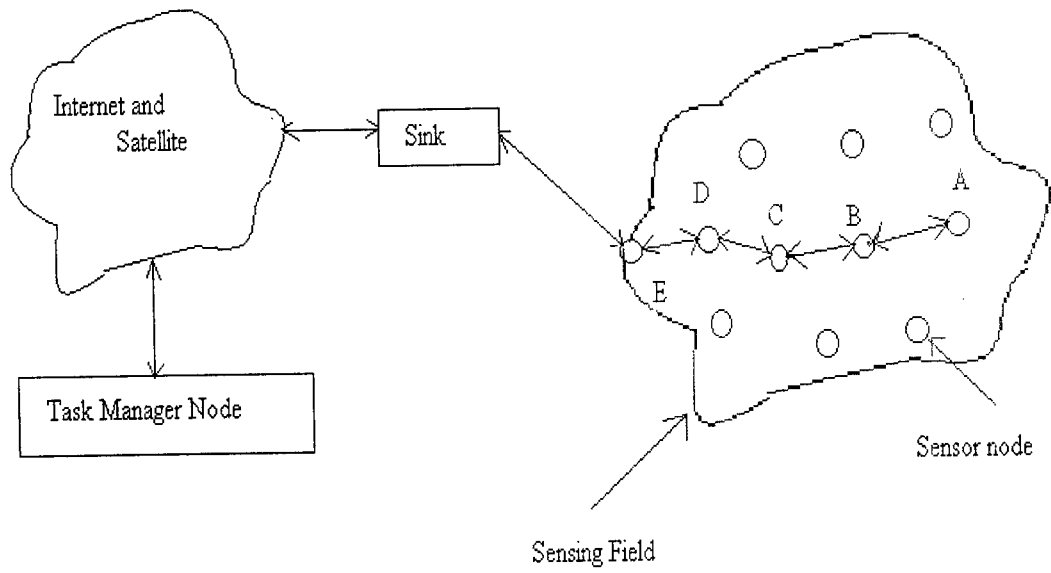


Figure 1.1.2[b] Sensor nodes scattered in sensor field

1.1.3. Components of sensor networks

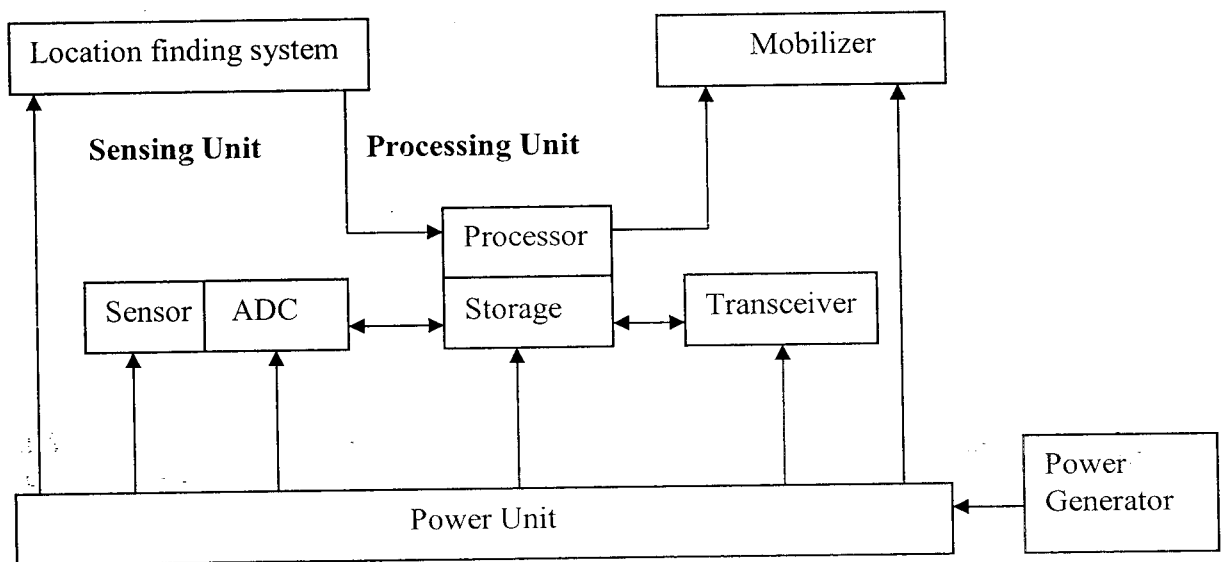


Figure 1.1.3[a].Components of sensor networks

- **Sensing unit**

Sensing units are usually composed of two subunits: sensors and analog to digital converters (ADCs). The analog signals produced by the sensors are converted to digital signals by the ADC, and then fed into the processing unit.

- **Processing Unit**

The processing unit which is generally associated with a small storage unit manages the procedures that make the sensor nodes collaborate with the other nodes to carry out the assigned sensing tasks.

- **Transceiver Unit**

A transceiver unit connects the nodes to the networks.

- **Power Unit**

One of the most important components of a sensor node is the power unit. Power units may be supported by a power scanning unit such as solar cells.

1.2 SENSOR NETWORK ARCHITECTURE

Communication in WSNs usually occurs in ad hoc manner, and shows similarities to wireless ad hoc networks. Likewise, WSNs are dynamic in the sense that radio range and network connectivity changes by time. Sensor nodes die and new sensor nodes may be added to the network. However, WSNs are more constrained, denser, and may suffer (or take advantage) of redundant information. WSN architectures are organized in hierarchical and distributed structures as shown in Fig. 1.2.

A Hierarchical WSNs (HWSN) is shown in Fig. 1.2[a], there is a hierarchy among the nodes based on their capabilities: base stations, cluster heads and sensor nodes. Base stations are many orders of magnitude more powerful than sensor nodes and cluster heads. A base station is typically a gateway to another network, a powerful data processing / storage center, or an access point for human interface. Base stations collect sensor readings, perform costly operations on behalf of sensor nodes and manage the network. In some applications, base stations are assumed to be trusted and temper resistant. Thus, they are used as key distribution centers.

Sensor nodes are deployed around one or more hop neighborhood of the base stations. They form a dense network where a cluster of sensors lying in a specific area may provide similar or close readings. Nodes with better resources, named as cluster heads, may be used collect and merge local traffic and send it to base stations. Transmission power of a base station is usually enough to reach all sensor nodes, but sensor nodes depend on the ad hoc communication to reach base stations. Thus, data flow in such networks can be: (i) pair-wise (unicast) among sensor nodes, (ii) group-wise (multicast) within a cluster of sensor nodes, and (iii) network-wise (broadcast) from base stations to sensor nodes.

A Distributed WSNs (DWSN) is shown in Fig. 1.2[b]. there is no fixed infrastructure, and network topology is not known prior to deployment. Sensor nodes are usually randomly scattered all over the target area. Once they are deployed, each sensor node scans its radio coverage area to figure out its neighbors. Data flow in DWSN is similar to data flow in HWSN with a difference that network-wise (broadcast) can be sent by every sensor nodes.

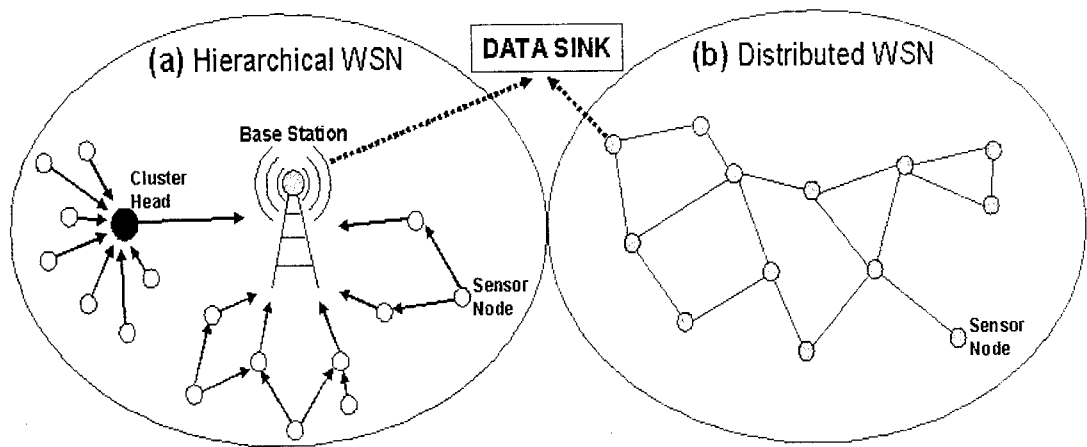


Figure 1.2 Network Models: Hierarchical and Distributed Wireless Sensor Networks.

1.3 CHARACTERISTICS OF WIRELESS SENSOR NETWORK

There are several characteristics available in wireless sensor networks and they are summarized as follows,

Unique characteristics of a WSN include:

- Limited power they can harvest or store
- Ability to withstand harsh environmental conditions
- Ability to cope with node failures
- Mobility of nodes
- Dynamic network topology
- Communication failures
- Heterogeneity of nodes
- Large scale of deployment
- Unattended operation

1.3.1 Sensor Node

Sensor nodes can be imagined as small computers, extremely basic in terms of their interfaces and their components. They usually consist of a processing unit with limited computational power and limited memory, sensors (including specific conditioning circuitry), a communication device (usually radio transceivers or alternatively optical), and a power source usually in the form of a battery. Other possible inclusions are energy harvesting modules, secondary ASICs, and possibly secondary communication devices (e.g. RS-232 or USB). The base stations are one or more distinguished components of the WSN with much more computational, energy and communication resources. They act as a gateway between sensor nodes and the end user.

1.4 APPLICATIONS OF SENSOR NETWORKS

Wireless Sensor Networks have a wide range of applications such as,

1. Military applications

- i. Monitoring friendly forces and equipment.
- ii. Battlefield surveillance.
- iii. Nuclear, biological and chemical attack detection.

2. Environmental Applications

- i. Forest fire detection.
- ii. Bio-complexity mapping of the environment.
- iii. Flood detection.

3. Health applications

- i. Tele-monitoring of human physiological data.
- ii. Tracking and monitoring doctors and patients inside a hospital.
- iii. Drug administration in hospitals.

In order to enable reliable and efficient observation and initiate right actions, physical phenomenon features should be reliably detected/estimated from the collective information provided by sensor nodes. Moreover, instead of sending the raw data to the nodes responsible for the fusion, sensor nodes use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data. Hence, these properties of WSN impose unique challenges for development of communication protocols in such architecture. The intrinsic properties of individual sensor nodes, pose additional challenges to the communication protocols in terms of energy consumption.

1.5 DATA COLLECTION METHODOLOGY

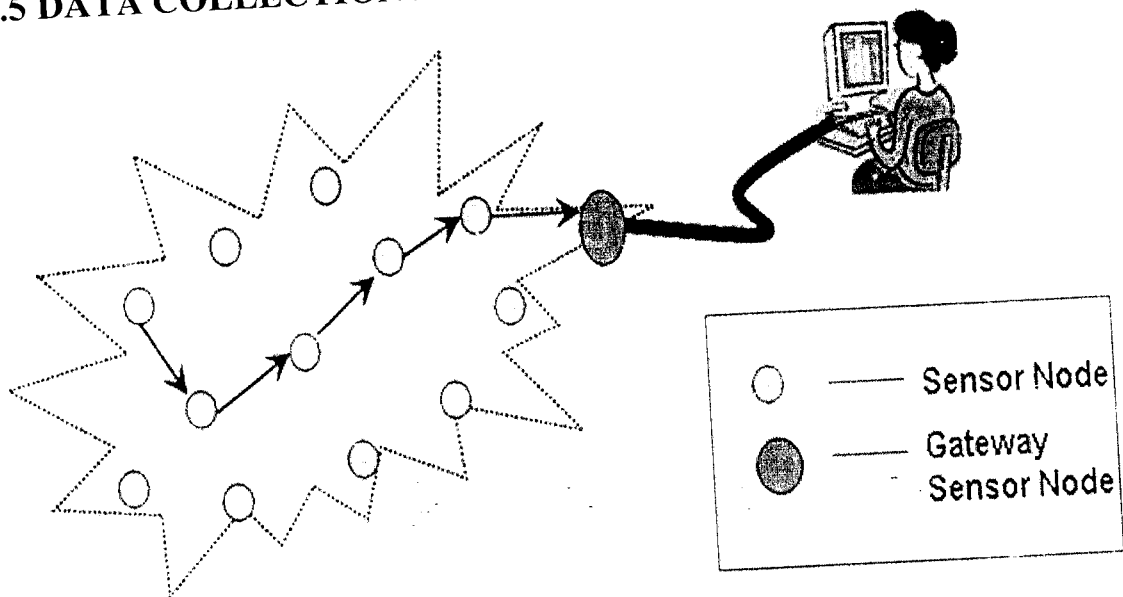


Figure 1.5[a].Data Collection in WSN

1.5.1 Event-based data collection

In event-based data collection the sensors are responsible for detecting and reporting events such as spotting moving targets. The event-based data collection is less demanding in terms of the amount of wireless communication, since local filtering is performed at the sensor nodes, and only events are propagated to the base node. In certain applications, the sensors may need to collaborate in order to detect events. Detecting complex events may necessitate nontrivial distributed algorithms that require the involvement of multiple sensor nodes. An inherent downside of the event based data collection is the impossibility of performing an in-depth analysis on the raw sensor readings since they are not extracted from the network in the first place.

1.5.2 Periodic data collection

Periodic updates are sent to the base node from the sensor network based on the most recent information sensed from the environment.

Two approaches are used to classify this,

1. In query-based data collection.
2. Long-standing queries are used to express user or application-specific information interests and these queries are installed “inside” the network. Most of the schemes following this approach support aggregate queries such as minimum, average, and maximum. These types of queries result in periodically generating an aggregate of the recent readings of all nodes. Although aggregation lends itself to simple implementations that enable the complete in-network processing of queries, it falls short in supporting holistic aggregates over sensor readings such as quintiles. Similar to the case of event-based data collection, the raw data is not extracted from the network and a complex data analysis that requires the integration of sensor readings from various nodes at various times cannot be performed with the in-network aggregation

1.6 CHALLENGES OF WIRELESS SENSOR NETWORKS

A sensor network design is influenced by many factors, which include,

- **Energy efficiency/system lifetime**

As sensor nodes are battery-operated, protocols must be energy-efficient to maximize system life time. System life time can be measured such as the time until half of the nodes die or by application-directed metrics, such as when the network stops providing the application with the desired information about the phenomena.

- **Fault Tolerance**

Some sensor nodes may fail or be blocked due to lack of power, have physical damage or environmental interference. The failure of sensor nodes should not affect the overall task of the sensor network. This is the reliability or fault tolerance issue. Fault tolerance is the ability to sustain sensor network functionalities without any interruption due to sensor node failures.

- **Scalability**

The number of sensor nodes deployed in studying a phenomenon may be in the order of hundreds or thousands. Depending on the application, the number may reach an extreme value of millions. The new schemes must be able to work with this number of nodes.

- **Production Costs**

Since the sensor networks consist of a large number of sensor nodes, the cost of a single node is very important to justify the overall cost of the networks. If the cost of the network is more expensive than deploying traditional sensors, then the sensor network is not cost-justified. As a result, the cost of each sensor node has to be kept low.

- **Environment**

Sensor nodes are densely deployed either very close or directly inside the phenomenon to be observed. Therefore, they usually work unattended in remote geographic areas. They may be working in busy intersections, in the interior of large

machinery, at the bottom of an ocean, inside a twister, on the surface of an ocean. They work under high pressure in the bottom of an ocean, in harsh environments such as debris or a battlefield, under extreme heat and cold such as in the nozzle of an aircraft engine or in arctic regions, and in an extremely noisy environment such as under intentional jamming.

- **Hardware Constraints**

A sensor node is made up of four basic components: a sensing unit, a processing unit, a transceiver unit and a power unit. Sensing units are usually composed of two subunits: sensors and analog to digital converters (ADCs). The analog signals produced by the sensors based on the observed phenomenon are converted to digital signals by the ADC, and then fed into the processing unit. The processing unit, which is generally associated with a small storage unit, manages the procedures that enable the sensor node collaborate with the other nodes to carry out the assigned sensing tasks. A transceiver unit connects the node to the network. One of the most important components of a sensor node is the power unit. Power units may be supported by a power scavenging unit such as solar cells. There are also other subunits, which are application dependent. Most of the sensor network routing techniques and sensing tasks require the knowledge of location with high accuracy.

- **Sensor Network Topology**

Sheer numbers of inaccessible and unattended sensor nodes, which are prone to frequent failures, make topology maintenance a challenging task. Hundreds to several thousands of nodes are deployed throughout the sensor field.

- **Pre-Deployment Phase**

Sensor nodes can be either thrown in mass or placed one by one in the sensor field. They can be deployed by dropping from a plane, delivering in an artillery shell, Rocket or missile, throwing by a catapult, placing in factory, and placing one by one either by a human or a robot. Although the sheer number of sensors and their unattended deployment usually preclude placing them according to a carefully engineered deployment plan, the schemes for initial deployment must reduce the installation cost, eliminate the need for any pre-organization and preplanning, increase the flexibility of arrangement, and promote self-organization and fault tolerance.

- **Post-Deployment Phase**

After deployment, topology changes are due to change in sensor nodes position, reach ability (due to jamming, noise, moving obstacles, etc.), available energy, malfunctioning, and task details. Sensor nodes may be statically deployed. However, device failure is a regular or common event due to energy depletion or destruction. It is also possible to have sensor networks with highly mobile nodes. Besides, sensor nodes and the network experience varying task dynamics, and they may be a target for deliberate jamming. Therefore, sensor network topologies are prone to frequent changes after deployment.

- **Re-Deployment of Additional Nodes Phase**

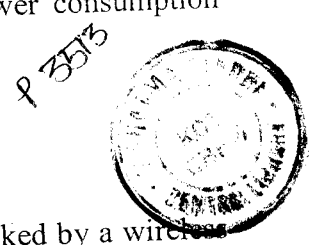
Additional sensor nodes can be re-deployed at any time to replace the malfunctioning nodes or due to changes in task dynamics. Addition of new nodes poses a need to re-organize the network. Coping with frequent topology changes in an ad hoc network that has myriads of nodes and very stringent power consumption constraints requires special routing protocols.

- **Transmission Media**

In a multi-hop sensor network, communicating nodes are linked by a wireless medium. These links can be formed by radio, infrared or optical media. To enable global operation of these networks, the chosen transmission medium must be available worldwide. One option for radio links is the use of *Industrial, Scientific and Medical* (ISM) bands, which offer license free communication in most countries.

- **Power Consumption**

The wireless sensor node, being a microelectronic device, can only be equipped with a limited power source. In some application scenarios, replenishment of power resources might be impossible. Sensor node lifetime, therefore, shows a strong dependence on battery lifetime. The malfunctioning of few nodes can cause significant topological changes and might require rerouting of packets and re-organization of the network. Hence, power conservation and power management take on additional importance. It is for these reasons that researchers are currently focusing



on the design of power aware protocols and algorithms for sensor networks. In sensor networks, power efficiency is an important performance metric, directly influencing the network lifetime. Application specific protocols can be designed by appropriately trading off other performance metrics such as delay and throughput with power efficiency. The main task of a sensor node in a sensor field is to detect events, perform quick local data processing, and then transmit the data. Power consumption can hence be divided into three domains: *sensing*, *communication*, and *data processing*.

CHAPTER 2

LITERATURE SURVEY

2.1 RANDOM KEY PREDISTRIBUTION SCHEMES FOR SENSOR NETWORKS.

Efficient bootstrapping of secure keys is of critical importance for secure sensor network applications. Local processing of sensor data requires secure node to node communication. They present three efficient random key predistribution schemes for solving the security bootstrapping problem in resource-constrained sensor networks. Each of these three schemes represents a different trade-off in the design space of random key protocols. The choice of which scheme is best for a given application will depend on which trade-off is the most appealing.

The q-composite scheme achieves significantly improved security under small scale attack at the cost of greater vulnerability to large scale attack. This increases the attacker's cost of mounting an attack since the option of harvesting a small number of keys in order to extract a random sample of the readings in the entire network is no longer appealing, thus forcing the attacker to perform a large scale node capture attack. The (2-hop) multipart reinforcement scheme improves security at the cost of network communication overhead.

Since the expected number of common neighbours is proportional to $1/n'$ (where n' is the expected number of neighbouring nodes), this scheme performs best when the deployment density is sparse relative to the communication radius of the nodes. It also presents the best characteristics when the variation in deployment density is low (i.e. nodes are regularly dispersed).

The random pair wise scheme has the best security properties of the three schemes. It possesses perfect resilience against node capture attacks as well as support for node-based revocation and resistance to node replication. The properties come with the trade-off that the maximum supported network size is not as large as the other schemes.

2.2 A KEY MANAGEMENT SCHEME IN DISTRIBUTED SENSOR NETWORKS USING ATTACK PROBABILITIES.

A variant of random key predistribution scheme for bootstrapping the clustered Distributed Sensor Network (DSN) when the probability of node compromise in different deployment regions is known priory. The cluster based hierarchical topology not only isolates the effect of node compromise into one specific subgroup and provides scalability for node and subgroup addition, but more importantly, it simplifies the design of key management scheme for the sensor networks.

With priory knowledge of the probability of node compromise, an effective scalable security mechanism that increases the resilience to the attacks for the sensor subgroups is designed. Simulation results demonstrated the substantial performance improvement (including the resilience and the fraction of compromised communications) compared to that of the prior schemes.

2.3 ENERGY-EFFICIENT COMMUNICATION PROTOCOL FOR WIRELESS MICRO SENSOR NETWORKS.

In this paper, the communication protocols, which can have significant impact on the overall energy dissipation of these networks is being explained. LEACH (Low-Energy Adaptive Clustering Hierarchy), a clustering-based protocol that utilizes randomized rotation of local cluster base stations (cluster-heads) to evenly distribute the energy load among the sensors in the network. LEACH uses localized coordination to enable scalability and robustness for dynamic networks, and incorporates data fusion into the routing protocol to reduce the amount of information that must be transmitted to the base station. Simulations show that LEACH can achieve as much as a factor of reduction in energy dissipation compared with conventional routing protocols. In addition, LEACH is able to distribute energy dissipation evenly throughout the sensors and improves the lifetime of the network.

2.4 SINGLE NETWORK-WIDE KEY

In the initialization phase of this technique, a single key is preloaded into all the nodes of the network. After deployment, every node in the network can use this key to encrypt and decrypt messages. Some of the advantages offered by this technique include minimal storage requirements and avoidance of complex protocols. Only a single key is to be stored in the nodes' memory and once deployed in the network, there is no need for a node to perform key discovery or key exchange since all the nodes in communication range can transfer messages using the key which they already share.

The main drawback is that compromise of a single node causes the compromise of the entire network through the shared key. This scheme counters several constraints with less computation and reduced memory use, but it fails in providing the basic requirements of a sensor network by making it easy for an adversary trying to attack.

2.5 SECURITY CONCERNS WITH RESPECT TO THIS DOCUMENT

The entirety of this document is security related. It discusses a security mechanism, Symmetric encryption, which is used to provide confidentiality of transmitted data. Encryption provides a means to prevent unauthorized disclosure of information. That is, if the information were not encrypted, a casual observer or an active attacker would be able to obtain the information.

2.6 POTENTIAL THREATS AND ATTACK SCENARIOS

If information's confidentiality is not protected using a mechanism such as encryption, it may be disclosed to unauthorized entities. In many cases this disclosure would not matter, but there are cases where it could result in the loss of proprietary data, loss of sensitive data, or loss of privacy data (e.g., human-crewed mission medical information). The information could be obtained by, for example, an eavesdropper listening to an RF transmission, a tap on a landline, or an unauthorized agency insider examining network traffic.

CHAPTER 3

ADVANCED ENCRYPTION STANDARD (AES)

3.1 INTRODUCTION

This standard specifies the **Rijndael** algorithm ([3] and [4]), a symmetric block cipher that can process **data blocks** of **128 bits**, using cipher **keys** with lengths of **128, 192, and 256 bits**. Rijndael was designed to handle additional block sizes and key lengths; however they are not adopted in this standard.

Throughout the remainder of this standard, the algorithm specified herein will be referred to as “the AES algorithm.” The algorithm may be used with the three different key lengths indicated above, and therefore these different “flavors” may be referred to as “AES-128”, “AES-192”, and “AES-256”.

This specification includes the following sections:

2. Definitions of terms, acronyms, and algorithm parameters, symbols, and functions;
3. Notation and conventions used in the algorithm specification, including the
Ordering and numbering of bits, bytes, and words;
4. Mathematical properties that is useful in understanding the algorithm;
5. Algorithm specification, covering the key expansion, encryption, and decryption
Routines;
6. Implementation issues, such as key length support, keying restrictions, and
Additional block/key/round sizes.

The standard concludes with several appendices that include step-by-step examples for Key Expansion and the Cipher, example vectors for the Cipher and Inverse Cipher, and a list of references.

3.2 DEFINITIONS

3.2.1 Glossary of Terms and Acronyms

The following definitions are used throughout this standard:

AES	Advanced Encryption Standard
Affine	A transformation consisting of multiplication by a matrix

	followed by
Transformation	the addition of a vector.
Array	An enumerated collection of identical entities (e.g., an array of bytes).
Bit	A binary digit having a value of 0 or 1.
Block	Sequence of binary bits that comprise the input, output, State, And Round Key. The length of a sequence is the number of bits it contains. Blocks are also interpreted as arrays of bytes.
Byte	A group of eight bits that is treated either as a single entity or as an array of 8 individual bits.
Cipher	Series of transformations that converts plaintext to cipher text using the Cipher Key.
Cipher Key	Secret, cryptographic key that is used by the Key Expansion routine to generate a set of Round Keys; can be pictured as a rectangular array of bytes, having four rows and Nk columns.
Ciphertext	Data output from the Cipher or input to the Inverse Cipher.
Inverse Cipher	Series of transformations that converts Ciphertext to plaintext using the Cipher Key.
Key Expansion	Routine used to generate a series of Round Keys from the Cipher Key.
Plaintext	Data input to the Cipher or output from the Inverse Cipher.
Rijndael	Cryptographic algorithm specified in this Advanced Encryption Standard (AES).
Round Key	Round keys are values derived from the Cipher Key using the Key Expansion routine; they are applied to the State in the Cipher and Inverse Cipher.
State	Intermediate Cipher result that can be pictured as a rectangular Array of bytes, having four rows and Nb columns.
S-box	Non-linear substitution table used in several byte substitution transformations and in the Key Expansion routine to perform a One-for-one substitution of a byte value.
Word	A group of 32 bits that is treated either as a single entity or as an array of 4 bytes.

3.2.2 Algorithm Parameters, Symbols, and Functions

The following algorithm parameters, symbols, and functions are used throughout this standard:

AddRoundKey()	Transformation in the Cipher and Inverse Cipher in which a Round Key is added to the State using an XOR operation. The length of a Round Key equals the size of the State (i.e., for $Nb = 4$, the Round Key length equals 128 bits/16 bytes).
InvMixColumns()	Transformation in the Inverse Cipher that is the inverse of MixColumns() .
InvShiftRows()	Transformation in the Inverse Cipher that is the inverse of ShiftRows() .
InvSubBytes()	Transformation in the Inverse Cipher that is the inverse of SubBytes() .
K	Cipher Key.
MixColumns()	Transformation in the Cipher that takes all of the columns of The State and mixes their data (independently of one another) to produce new columns.
Nb	Number of columns (32-bit words) comprising the State. For this standard, $Nb = 4$.
Nk	Number of 32-bit words comprising the Cipher Key. For this standard, $Nk = 4, 6, \text{ or } 8$.
Nr	Number of rounds, which is a function of Nk and Nb (which is fixed). For this standard, $Nr = 10, 12, \text{ or } 14$.
Rcon[]	The round constant word array.
RotWord()	Function used in the Key Expansion routine that takes a four-Byte word and performs a cyclic permutation.
ShiftRows()	Transformation in the Cipher that processes the State by cyclically shifting the last three rows of the State by different offsets.
SubBytes()	Transformation in the Cipher that processes the State using a Nonlinear byte substitution table (S-box) that operates on each of the State bytes independently.
SubWord()	Function used in the Key Expansion routine that takes a four-Byte input word and applies an S-box to each of the four

- bytes to produce an output word.
- XOR Exclusive-OR operation.
- \oplus Exclusive-OR operation.
- \otimes Multiplication of two polynomials (each with degree < 4)
Modulo $x^4 + 1$.
- Finite field multiplication.

3.3 NOTATION AND CONVENTIONS

3.3.1 Inputs and Outputs

The **input** and **output** for the AES algorithm each consist of **sequences of 128 bits** (digits with values of 0 or 1). These sequences will sometimes be referred to as **blocks** and the number of bits they contain will be referred to as their length. The **Cipher Key** for the AES algorithm is a **sequence of 128, 192 or 256 bits**. Other input, output and Cipher Key lengths are not permitted by this standard.

The bits within such sequences will be numbered starting at zero and ending at one less than the sequence length (block length or key length). The number i attached to a bit is known as its index and will be in one of the ranges $0 \leq i < 128$, $0 \leq i < 192$ or $0 \leq i < 256$ depending on the block length and key length (specified above).

3.3.2 Bytes

The basic unit for processing in the AES algorithm is a **byte**, a sequence of eight bits treated as a single entity. The input, output and Cipher Key bit sequences described in Sec. 3.1 are processed as arrays of bytes that are formed by dividing these sequences into groups of eight contiguous bits to form arrays of bytes (see Sec. 3.3). For an input, output or Cipher Key denoted by a , the bytes in the resulting array will be referenced using one of the two forms, a_n or $a[n]$, where n will be in one of the following ranges:

Key length = 128 bits, $0 \leq n < 16$;

Key length = 192 bits, $0 \leq n < 24$;

Key length = 256 bits, $0 \leq n < 32$.

Block length = 128 bits, $0 \leq n < 16$;

All byte values in the AES algorithm will be presented as the concatenation of its individual bit values (0 or 1) between braces in the order $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$. These bytes are interpreted as finite field elements using a polynomial representation:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = \sum_{i=0}^7 b_i x^i. \quad (3.1)$$

For example, $\{01100011\}$ identifies the specific finite field element $x_6 + x_5 + x + 1$.

It is also convenient to denote byte values using hexadecimal notation with each of two groups of four bits being denoted by a single character as in Fig. 3.3.2[A].

Bit Pattern	Character
0000	0
0001	1
0010	2
0011	3

Bit Pattern	Character
0100	4
0101	5
0110	6
0111	7

Bit Pattern	Character
1000	8
1001	9
1010	a
1011	b

Bit Pattern	Character
1100	c
1101	d
1110	e
1111	f

Figure 3.3.2[a] Hexadecimal representation of bit patterns.

Hence the element $\{01100011\}$ can be represented as $\{63\}$, where the character denoting the four-bit group containing the higher numbered bits is again to the left. Some finite field operations involve one additional bit (b_8) to the left of an 8-bit byte. Where this extra bit is present, it will appear as ‘ $\{01\}$ ’ immediately preceding the 8-bit byte; for example, a 9-bit sequence will be presented as $\{01\}\{1b\}$.

3.3.3 Arrays of Bytes

Arrays of bytes will be represented in the following form:

$$a_0 a_1 a_2 \dots a_{15}$$

The bytes and the bit ordering within bytes are derived from the 128-bit input sequence

$$input_0 \ input_1 \ input_2 \ \dots \ input_{126} \ input_{127}$$

as follows:

$$\begin{aligned}
 a_0 &= \{input_0, input_1, \dots, input_7\}; \\
 a_1 &= \{input_8, input_9, \dots, input_{15}\}; \\
 &\vdots \\
 a_{15} &= \{input_{120}, input_{121}, \dots, input_{127}\}.
 \end{aligned}$$

The pattern can be extended to longer sequences (i.e., for 192- and 256-bit keys), so that, in general,

$$a_n = \{input_{8n}, input_{8n+1}, \dots, input_{8n+7}\}. \quad (3.2)$$

Taking Sections 3.3.2 and 3.3.3 together, Fig. 3.3.3[a] shows how bits within each byte are numbered.

Input bit sequence	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	...
Byte number	0							1							2							...			
Bit numbers in byte	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	...

Figure 3.3.3[a] Indices for Bytes and Bits.

3.3.4 The State

Internally, the AES algorithm's operations are performed on a two-dimensional array of bytes called the **State**. The State consists of four rows of bytes, each containing Nb bytes, where Nb is the block length divided by 32. In the State array denoted by the symbol s , each individual byte has two indices, with its row number r in the range $0 \leq r < 4$ and its column number c in the range $0 \leq c < Nb$. This allows an individual byte of the State to be referred to as either $s_{r,c}$ or $s[r,c]$. For this standard, $Nb=4$, i.e., $0 \leq c < 4$.

At the start of the Cipher and Inverse Cipher described in Sec. 3.5, the input – the array of bytes $in_0, in_1, \dots, in_{15}$ – is copied into the State array as illustrated in Fig. 3.3.4[a]. The Cipher or Inverse Cipher operations are then conducted on this State array, after which its final value is copied to the output – the array of bytes $out_0, out_1, \dots, out_{15}$.

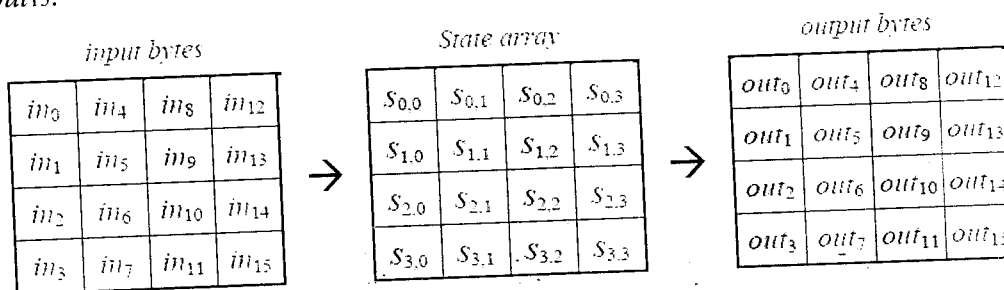


Figure 3.3.4[a] State array input and output.

Hence, at the beginning of the Cipher or Inverse Cipher, the input array, in , is copied to the State array according to the scheme:

$$s[r, c] = in[r + 4c] \quad \text{for } 0 \leq r < 4 \text{ and } 0 \leq c < Nb, \quad (3.3)$$

and at the end of the Cipher and Inverse Cipher, the State is copied to the output array *out* as follows:

$$out[r + 4c] = s[r, c] \quad \text{for } 0 \leq r < 4 \text{ and } 0 \leq c < Nb. \quad (3.4)$$

3.3.5 The State as an Array of Columns

The four bytes in each column of the State array form 32-bit **words**, where the row number *r* provides an index for the four bytes within each word. The state can hence be interpreted as a one-dimensional array of 32 bit words (columns), $w_0 \dots w_3$, where the column number *c* provides an index into this array. Hence, for the example in Fig. 3.3.4[a] the State can be considered as an array of four words, as follows:

$$\begin{aligned} w_0 &= s_{0,0} s_{1,0} s_{2,0} s_{3,0} \\ w_1 &= s_{0,1} s_{1,1} s_{2,1} s_{3,1} \\ w_2 &= s_{0,2} s_{1,2} s_{2,2} s_{3,2} \\ w_3 &= s_{0,3} s_{1,3} s_{2,3} s_{3,3}. \end{aligned} \quad (3.5)$$

3.4 MATHEMATICAL PRELIMINARIES

All bytes in the AES algorithm are interpreted as finite field elements using the notation introduced in Sec. 3.3.2. Finite field elements can be added and multiplied, but these operations are different from those used for numbers. The following subsections introduce the basic mathematical concepts needed for Sec. 3.5.

3.5 ALGORITHM SPECIFICATION

For the AES algorithm, **the length of the input block, the output block and the State is 128 bits**. This is represented by $Nb = 4$, which reflects the number of 32-bit words (number of columns) in the State.

For the AES algorithm, **the length of the Cipher Key, *K*, is 128, 192, or 256 bits**. The key length is represented by $Nk = 4, 6, \text{ or } 8$, which reflects the number of 32-bit words (number of columns) in the Cipher Key.

For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by Nr , where $Nr = 10$ when $Nk = 4$, $Nr = 12$ when $Nk = 6$, and $Nr = 14$ when $Nk = 8$.

The only Key-Block-Round combinations that conform to this standard are given in Fig.3.5 For implementation issues relating to the key length, block size and number of rounds, see Sec.3.6.3

	Key Length (<i>Nk words</i>)	Block Size (<i>Nb words</i>)	Number of Rounds (<i>Nr</i>)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Figure 3.5 Key-Block-Round Combinations.

For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations: 1) byte substitution using a substitution table (S-box), 2) shifting rows of the State array by different offsets, 3) mixing the data within each column of the State array, and 4) adding a Round Key to the State. These transformations (and their inverses) are described in Sec. 3.5.1.1-3.5.1.4 and 3.5.3.1-3.5.3.4.

The Cipher and Inverse Cipher are described in Sec. 3.5.1 and Sec. 3.5.3, respectively, while the Key Schedule is described in Sec. 3.5.2.

3.5.1 Cipher

At the start of the Cipher, the input is copied to the State array using the conventions described in Sec. 3.3.4. After an initial Round Key addition, the State array is transformed by implementing a round function 10, 12, or 14 times (depending on the key length), with the final round differing slightly from the first $Nr - 1$ rounds. The final State is then copied to the output as described in Sec. 3.3.4.

The round function is parameterized using a key schedule that consists of a one-dimensional array of four-byte words derived using the Key Expansion routine described in Sec. 3.5.2.

Key size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext block size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of rounds	10	12	14
Round key size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded key size (words/bytes)	44/176	52/208	60/240

Figure 3.5.1[a] AES parameters

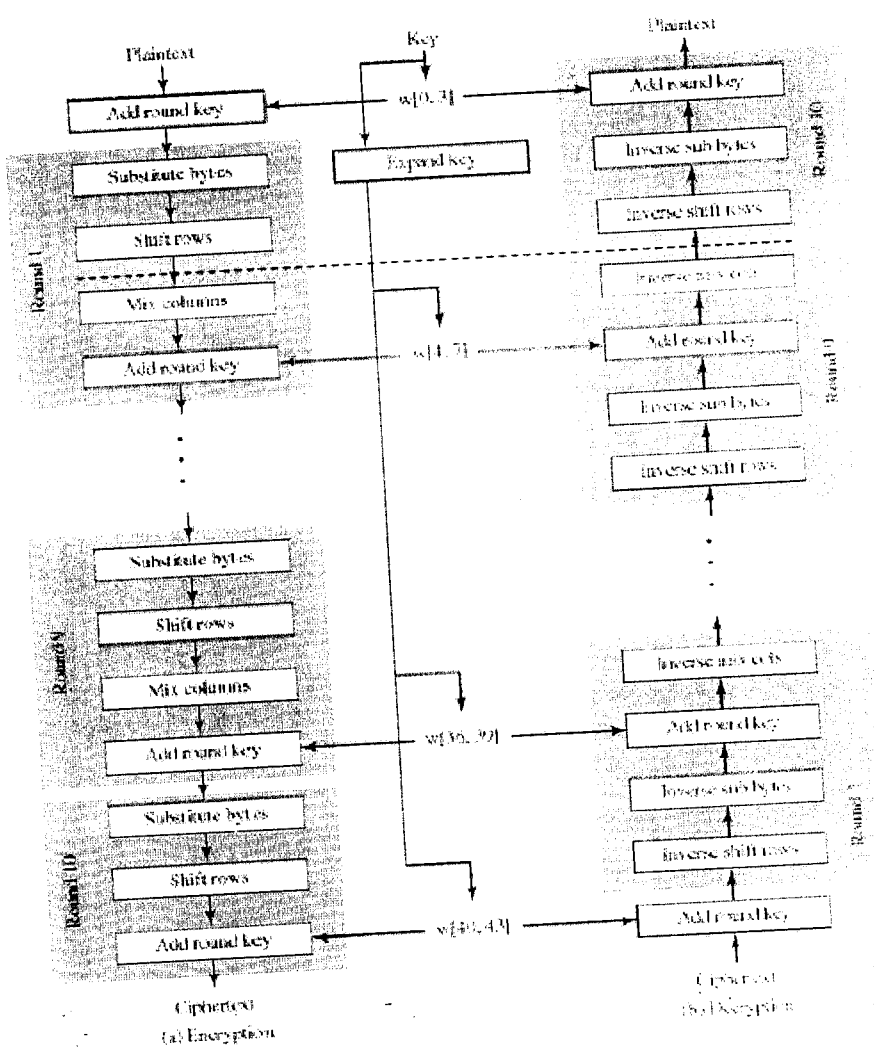


Figure 3.5.1[b] AES Encryption and Decryption

3.5.1.1 SubBytes Transformation

The **SubBytes()** transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box). This S-box (Fig. 3.5.1.1[a]), which is invertible, is constructed by composing two transformations:

1. Take the multiplicative inverse in the finite field $GF(2^8)$, described in Sec. 3.4.2; the element $\{00\}$ is mapped to itself.
2. Apply the following affine transformation (over $GF(2^8)$):

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \quad (5.1)$$

for $0 \leq i < 8$, where b_i is the i^{th} bit of the byte, and c_i is the i^{th} bit of a byte c with the value $\{63\}$ or $\{011100011\}$. Here and elsewhere, a prime on a variable (e.g., b') indicates that the variable is to be updated with the value on the right.

In matrix form, the affine transformation element of the S-box can be expressed as:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (5.2)$$

Figure 6 illustrates the effect of the **SubBytes()** transformation on the State.

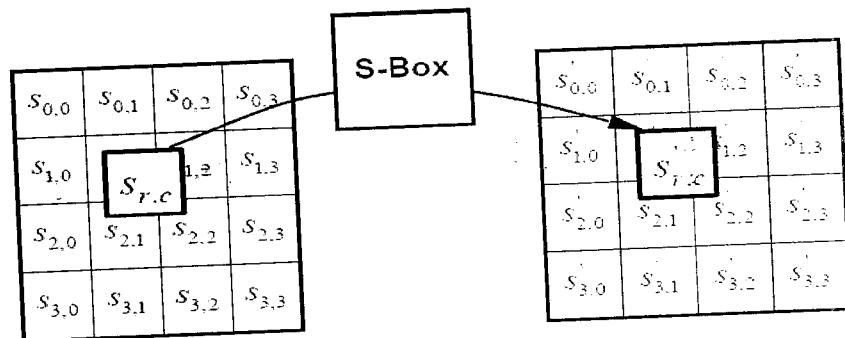


Figure 3.5.1.1[a] SubBytes() applies the S-box to each byte of the State.

The S-box used in the **SubBytes()** transformation is presented in hexadecimal form in Fig.3.5.1.1[b]. For example, if $S_{1,1} = \{53\}$, then the substitution value would be determined by the intersection of the row with index '5' and the column with index '3' in Fig. 3.5.1.1[b]. This would result in $S'_{1,1}$ having a value of {ed}.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 3.5.1.1[b] S-box: substitution values for the byte xy (in hexadecimal format).

3.5.1.2 ShiftRows Transformation

In the **ShiftRows()** transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row, $r = 0$, is not shifted. Specifically, the **ShiftRows()** transformation proceeds as follows:

$$S'_{r,c} = S_{r,(c+shift(r,Nb)) \bmod Nb} \quad \text{for } 0 < r < 4 \quad \text{and} \quad 0 \leq c < Nb. \quad (5.3)$$

Where the shift value $shift(r,Nb)$ depends on the row number, r , as follows (recall that $Nb = 4$):

$$shift(1,4) = 1; \quad shift(2,4) = 2; \quad shift(3,4) = 3. \quad (5.4)$$

This has the effect of moving bytes to “lower” positions in the row (i.e., lower values of c in a given row), while the “lowest” bytes wrap around into the “top” of the row (i.e., higher values of c in a given row).

Figure 3.5.1.2[a] illustrates the **ShiftRows()** transformation.

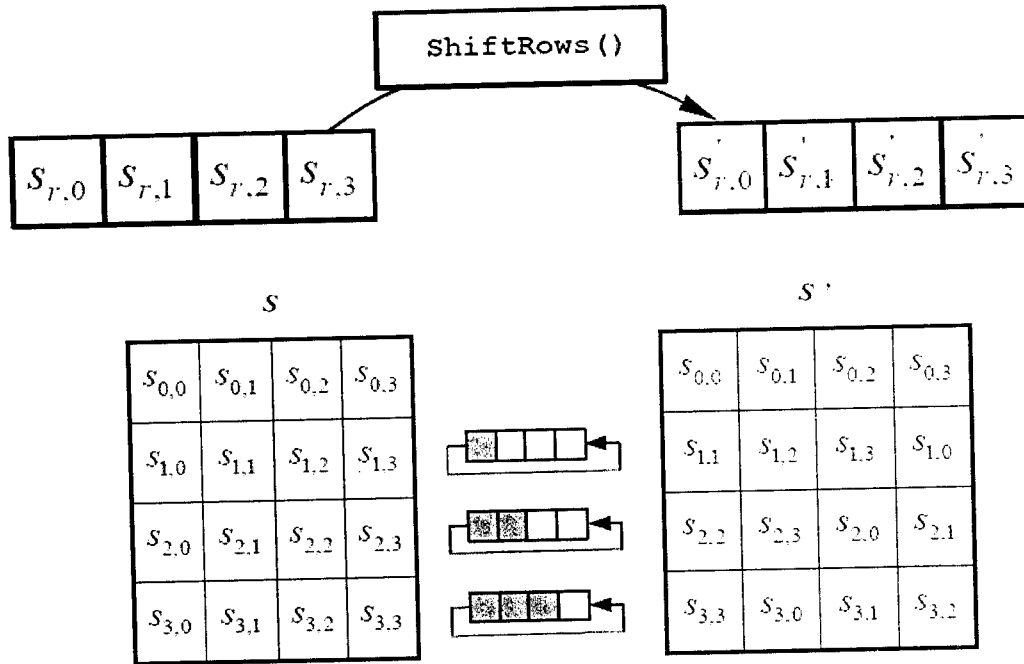


Figure 3.5.1.2[a] ShiftRows() cyclically shifts the last three rows in the State.

3.5.1.3 MixColumns Transformation

The **MixColumns()** transformation operates on the State column-by-column, treating each column as a four-term polynomial as described in Sec. 3.4.3. The columns are considered as polynomials over $\text{GF}(2^8)$ and multiplied modulo $x^4 + 1$ with a fixed polynomial $a(x)$, given by

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}. \quad (5.5)$$

As described in Sec. 3.4.3, this can be written as a matrix multiplication. Let

$$s'(x) = a(x) \otimes s(x) :$$

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \leq c < Nb. \quad (5.6)$$

As a result of this multiplication, the four bytes in a column are replaced by the following:

$$\begin{aligned}
s'_{0,c} &= (\{02\} \cdot s_{0,c}) \oplus (\{03\} \cdot s_{1,c}) \oplus s_{2,c} \oplus s_{3,c} \\
s'_{1,c} &= s_{0,c} \oplus (\{02\} \cdot s_{1,c}) \oplus (\{03\} \cdot s_{2,c}) \oplus s_{3,c} \\
s'_{2,c} &= s_{0,c} \oplus s_{1,c} \oplus (\{02\} \cdot s_{2,c}) \oplus (\{03\} \cdot s_{3,c}) \\
s'_{3,c} &= (\{03\} \cdot s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \cdot s_{3,c}).
\end{aligned}$$

Figure 3.5.1.3[a] illustrates the **MixColumns()** transformation.

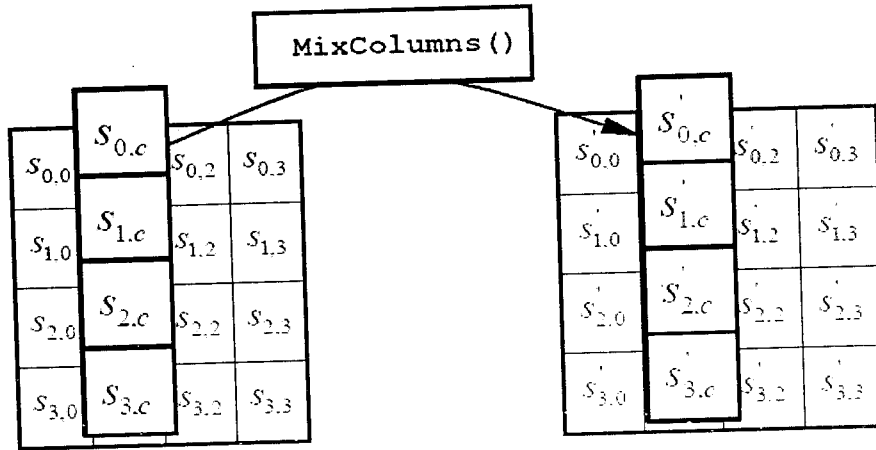


Figure 3.5.1.3[a] MixColumns() operates on the State column-by-column.

3.5.1.4 AddRoundKey Transformation

In the **AddRoundKey()** transformation, a Round Key is added to the State by a simple bitwise XOR operation. Each Round Key consists of Nb words from the key schedule (described in Sec.3.5.2). Those Nb words are each added into the columns of the State, such that

$$[s'_{0,c} \cdot s'_{1,c} \cdot s'_{2,c} \cdot s'_{3,c}] = [s_{0,c} \cdot s_{1,c} \cdot s_{2,c} \cdot s_{3,c}] \oplus [w_{round \cdot Nb - c}] \quad \text{for } 0 \leq c < Nb. \quad (5.7)$$

where $[w_i]$ are the key schedule words described in Sec. 3.5.2, and $round$ is a value in the range $0 \leq round \leq Nr$. In the Cipher, the initial Round Key addition occurs when $round = 0$. The application of the **AddRoundKey()** transformation to the Nr rounds of the Cipher occurs when $1 \leq round \leq Nr$.

The action of this transformation is illustrated in Fig.3.5.1.4 [a], where $l = round \cdot Nb$. The byte address within words of the key schedule was described in Sec. 3.3.1.

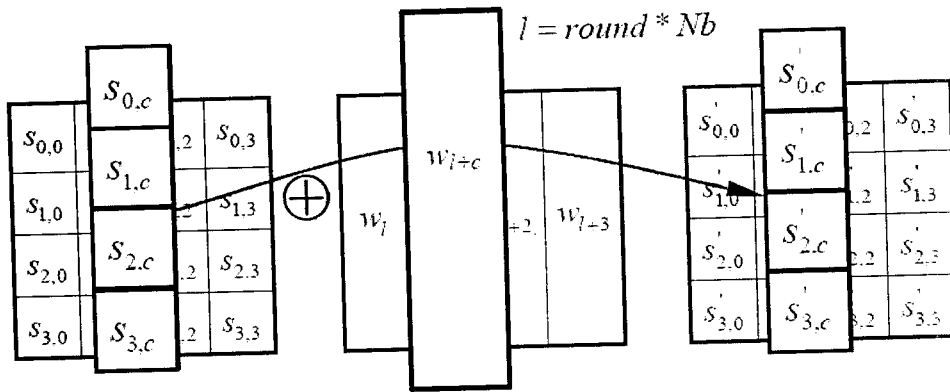


Figure 3.5.1.4[a] AddRoundKey() XORs each column of the State with a word from the key schedule.

3.5.2 Key Expansion

The AES algorithm takes the Cipher Key, \mathbf{K} , and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of Nb ($Nr + 1$) words: the algorithm requires an initial set of Nb words, and each of the Nr rounds requires Nb words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted $[w_i]$, with i in the range $0 \leq i < Nb(Nr + 1)$.

SubWord() is a function that takes a four-byte input word and applies the S-box (Sec. 3.5.1.1, Fig. 3.5.1.1[a]) to each of the four bytes to produce an output word. The function **RotWord()** takes a word $[a_0, a_1, a_2, a_3]$ as input, performs a cyclic permutation, and returns the word $[a_1, a_2, a_3, a_0]$. The round constant word array, **Rcon[i]**, contains the values given by $[x^{i-1}, \{00\}, \{00\}, \{00\}]$, with x^{i-1} being powers of x (x is denoted as $\{02\}$) in the field $GF(2^8)$.

The first Nk words of the expanded key are filled with the Cipher Key. Every following word, $\mathbf{w}[i]$, is equal to the XOR of the previous word, $\mathbf{w}[i-1]$, and the word Nk positions earlier, $\mathbf{w}[i-Nk]$. For words in positions that are a multiple of Nk , a transformation is applied to $\mathbf{w}[i-1]$ prior to the XOR, followed by an XOR with a round constant, **Rcon[i]**. This transformation consists of a cyclic shift of the bytes in a word (**RotWord()**), followed by the application of a table lookup to all four bytes of the word (**SubWord()**).

It is important to note that the Key Expansion routine for 256-bit Cipher Keys ($Nk = 8$) is slightly different than for 128- and 192-bit Cipher Keys. If $Nk = 8$ and $i - 4$ is a multiple of Nk , then **SubWord()** is applied to $w[i - 1]$ prior to the XOR.

3.5.3 Inverse Cipher

The Cipher transformations in Sec. 3.5.1 can be inverted and then implemented in reverse order to produce a straightforward Inverse Cipher for the AES algorithm. The individual transformations used in the Inverse Cipher - **InvShiftRows()**, **InvSubBytes()**, **InvMixColumns()**, and **AddRoundKey()** - process the State and are described in the following subsections.

3.5.3.1 InvShiftRows Transformation

InvShiftRows() is the inverse of the **ShiftRows()** transformation. The bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row, $r = 0$, is not shifted. The bottom three rows are cyclically shifted by $Nb - shift(r, Nb)$ bytes, where the shift value $shift(r, Nb)$ depends on the row number, and is given in equation (5.4) (see Sec. 3.5.1.2).

Specifically, the **InvShiftRows()** transformation proceeds as follows:

$$S_{r, (c - shift(r, Nb)) \bmod Nb} = S_{r, c} \quad \text{for } 0 < r < 4 \quad \text{and} \quad 0 \leq c < Nb \quad (5.8)$$

Figure 3.5.3.1[a] illustrates the **InvShiftRows()** transformation.

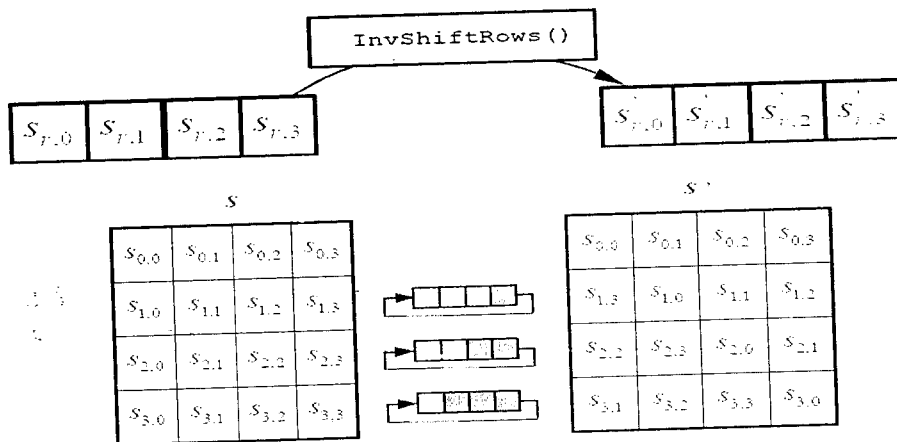


Figure 3.5.3.1[a] **InvShiftRows()** cyclically shifts the last three rows in the State.

3.5.3.2 InvSubBytes Transformation

InvSubBytes() is the inverse of the byte substitution transformation, in which the inverse Sbox is applied to each byte of the State. This is obtained by applying the inverse of the affine transformation (3.5.1) followed by taking the multiplicative inverse in $GF(2^8)$.

The inverse S-box used in the **InvSubBytes()** transformation is presented in

Fig.3.5.3.2[a]

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Figure 3.5.3.2[a] Inverse S-box: substitution values for the byte xy (in Hexadecimal format).

3.5.3.3 InvMixColumns Transformation

InvMixColumns() is the inverse of the **MixColumns()** transformation. **InvMixColumns()** operates on the State column-by-column, treating each column as a four-term polynomial. The columns are considered as polynomials over $GF(2^8)$ and multiplied modulo $x^4 + 1$ with a fixed polynomial $a^{-1}(x)$, given by

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}. \quad (5.9)$$

This can be written as a matrix multiplication. Let

$$s'(x) = a^{-1}(x) \otimes s(x):$$

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \leq c < Nb. \quad (5.10)$$

As a result of this multiplication, the four bytes in a column are replaced by the following:

$$\begin{aligned}
 s'_{0,c} &= (\{0e\} \cdot s_{0,c}) \oplus (\{0b\} \cdot s_{1,c}) \oplus (\{0d\} \cdot s_{2,c}) \oplus (\{09\} \cdot s_{3,c}) \\
 s'_{1,c} &= (\{09\} \cdot s_{0,c}) \oplus (\{0e\} \cdot s_{1,c}) \oplus (\{0b\} \cdot s_{2,c}) \oplus (\{0d\} \cdot s_{3,c}) \\
 s'_{2,c} &= (\{0d\} \cdot s_{0,c}) \oplus (\{09\} \cdot s_{1,c}) \oplus (\{0e\} \cdot s_{2,c}) \oplus (\{0b\} \cdot s_{3,c}) \\
 s'_{3,c} &= (\{0b\} \cdot s_{0,c}) \oplus (\{0d\} \cdot s_{1,c}) \oplus (\{09\} \cdot s_{2,c}) \oplus (\{0e\} \cdot s_{3,c})
 \end{aligned}$$

3.5.3.4 Inverse of the AddRoundKey Transformation

AddRoundKey(), which was described in Sec. 3.5.1.4, is its own inverse, since it only involves an application of the XOR operation.

3.5.3.5 Equivalent Inverse Cipher

In the straightforward Inverse Cipher presented in Sec. 3.5., the sequence of the transformations differs from that of the Cipher, while the form of the key schedules for encryption and decryption remains the same. However, several properties of the AES algorithm allow for an Equivalent Inverse Cipher that has the same sequence of transformations as the Cipher (with the transformations replaced by their inverses). This is accomplished with a change in the key schedule.

The two properties that allow for this Equivalent Inverse Cipher are as follows:

1. The **SubBytes()** and **ShiftRows()** transformations commute; that is, a **SubBytes()** transformation immediately followed by a **ShiftRows()** transformation is equivalent to a **ShiftRows()** transformation immediately followed by a **SubBytes()** transformation. The same is true for their inverses, **InvSubBytes()** and **InvShiftRows()**.
2. The column mixing operations - **MixColumns()** and **InvMixColumns()** - are linear with respect to the column input, which means $\text{InvMixColumns}(\text{state XOR Round Key}) = \text{InvMixColumns}(\text{state}) \text{ XOR } \text{InvMixColumns}(\text{Round Key})$.

These properties allow the order of **InvSubBytes()** and **InvShiftRows()** transformations to be reversed. The order of the **AddRoundKey()** and **InvMixColumns()** transformations can also be reversed, provided that the columns (words) of the decryption key schedule are modified using the **InvMixColumns()** transformation.

The equivalent inverse cipher is defined by reversing the order of the **InvSubBytes()** and **InvShiftRows()** transformations shown in Fig. 12, and by reversing the order of the **AddRoundKey()** and **InvMixColumns()** transformations used in the “round loop” after first modifying the decryption key schedule for $round = 1$ to $Nr-1$ using the **InvMixColumns()** transformation. The first and last Nb words of the decryption key schedule shall *not* be modified in this manner.

CHAPTER 4

PROPOSED SCHEME

The main motivation behind the idea of a dynamic cryptographic algorithm for WSN is to make the vital data and information stored in each of the nodes secure enough to allow the use of simpler and more energy efficient protocols for data transmission and communication over the network. As discussed in the subsequent sections, this method helps in avoiding the drawbacks, complexity and the cost involved in the other protocols while providing an acceptable level of security both for data transmission and data storage at the sensor nodes. In the given proposal the data present in each of the nodes has been divided into two parts:-

- **Hard – coded information**

The codes of the programs and other data embedded into the nodes before deployment, which are used to process the sensed information and to perform other operation related to data transmission, security and control. These hard coded Information are mutable.

- **Sensed information**

The information gathered by the sensor node from the environment it is monitoring.

4.1 DEMERITS OF AES

- ✓ It has large block size, so it requires the key with more number of bits.
- ✓ It uses same key for both Encryption and Decryption, so the hacking possibility is high.
- ✓ Key expansion mechanism is very complex.
- ✓ The Encryption algorithm differs from Decryption algorithm. Extra space is needed for the Decryption.

4.2 FLOW CHART

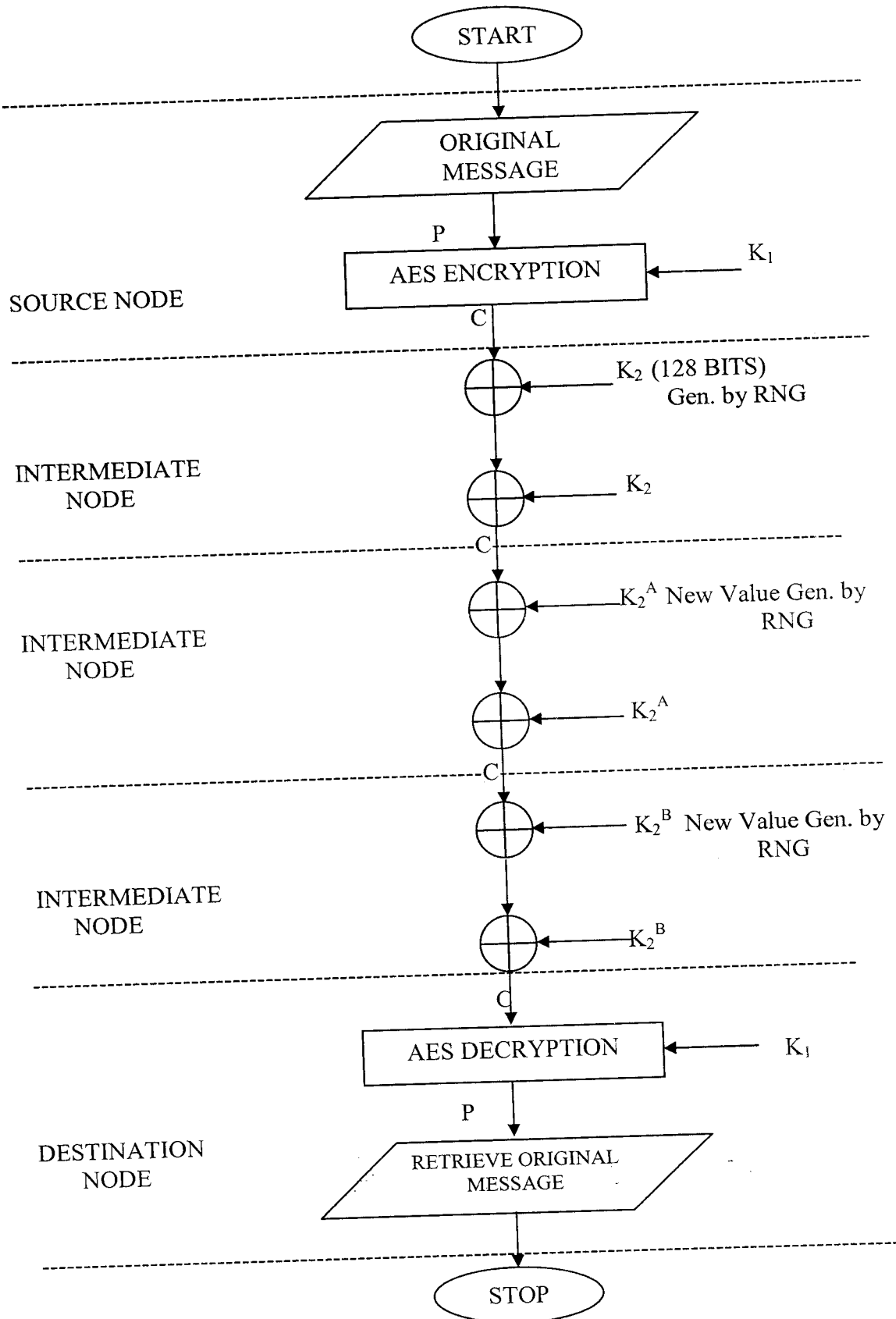


Figure 4.2[a] flow chart

4.3 CONCEPT

The proposed concept is as follows:-

- All the nodes will be embedded with a single networkwide shared key, say $K1$ ($K1$ comes under the category of hard-coded information). $K1$ will be used for message transmission using an AES symmetric encryption protocol.
- All nodes will also contain the sensed information from the environment they are monitoring. Since the sensor node does not process the sensed data itself, so we can encrypt this data as soon as it is gathered from the environment using the key $K1$ and the cryptographic protocol used in message transmission.
- All the intermediate nodes contain a second key $K2$, generated by RNG (Random Number Generator).
- $K2$ will be used for encrypting the received data from the neighbour node by using simple logical, invertible operation(s) such as XOR so that computation does not consume much energy].
- A resourceful attacker such as an attacker equipped with a laptop can easily guess the key $K2$ thereby exposing $K1$, thus gaining access to the sensed information that was encrypted using $K1$ and thus the entire network gets compromised since $K1$ is the only network-wide shared key used for message Encryption and Decryption only at source and Destination nodes. So to address this security threat $K2$ is made dynamic by changing its value periodically by using RNG. So that by the time the attacker comes close to guessing the value of $K2$, it gets changes as a result rendering $K2$ unsure for the attacker.
- All the intermediate nodes first checks its address with the received packet destination address either match or mismatch, if the node address mismatched that particular nodes performs the double time XOR operation with the same key $K2$ and forward to the neighbour node.

- If the node address matched with the received packet destination address then that particular node called as a Destination node, So it must performs Decryption process and retrieve the original plaintext.

4.4 ADVANTAGES OF DYNAMIC KEY

- ✓ The Dynamic key method is highly secured than AES.
- ✓ The attacker won't be able to guess the Dynamic key value because it is generated from RNG.
- ✓ All the intermediate nodes perform XOR operation twice with same Dynamic key value, which increases the security.
- ✓ Since all the intermediate nodes perform simple XOR operation, it utilises less resource.

CHAPTER 5

ROUTING PROTOCOL

5.1 DSDV

DS4.2 FLOW CHART

DV is one of the protocols for ad hoc mobile networks and its inception contributed to the evolution of numerous additional routing protocols. DSDV is an elementary and moderately less complex protocol which is suited fit for less dense network i.e. its targeted to function exquisitely on small node density. DSDV does Work, essentially, by sharing routing information with neighbouring nodes, which is stacked away by each node in the form of tables. DSDV, though simple posses a few shortcomings, since the routing updates are sudden and sporadic, even a minor change in network topology whether logical or physical drives the protocol to exchange the full routing table across the complete network. This holds the network busy in continuation. Which is a heavy processing overhead time for sending even small amounts of data.

Table 5.1[a] Average and Total Delay between AODV & DSDV

<i>Name of Protocol</i>	<i>Average Delay</i>	<i>Total Delay</i>
AODV	0.003071	0.872216
DSDV	0.004552	2.960383

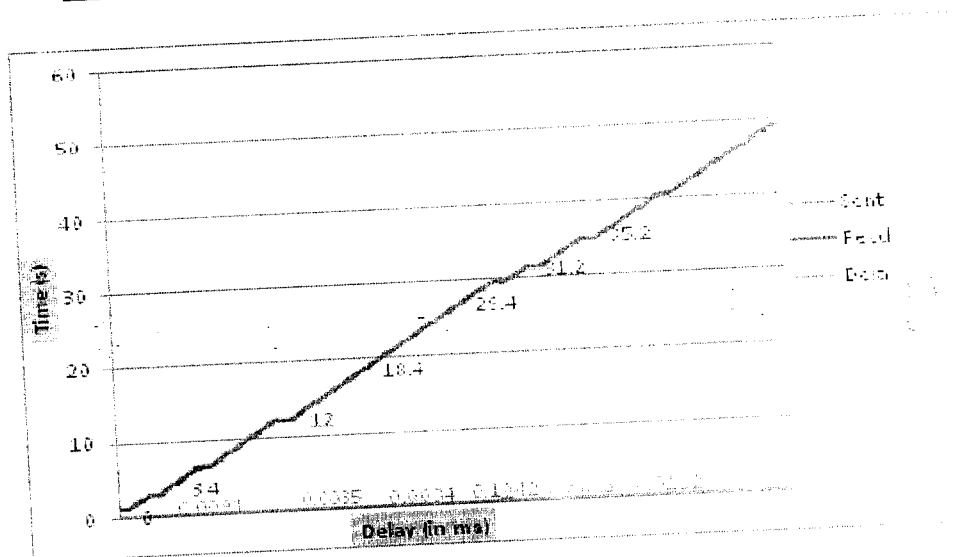


Figure 5.1[b] AODV Delay

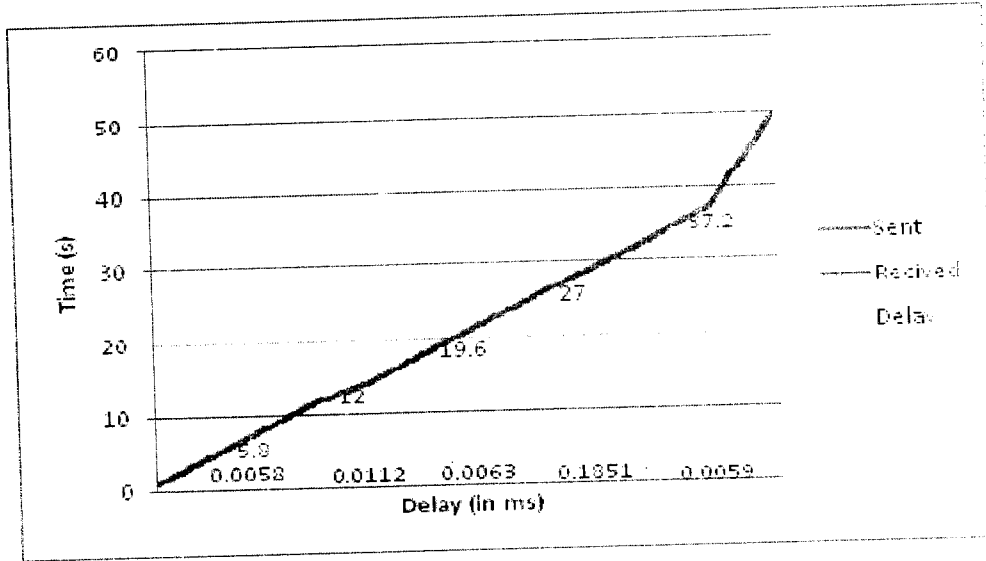


Figure 5.1[c] DSDV Delay

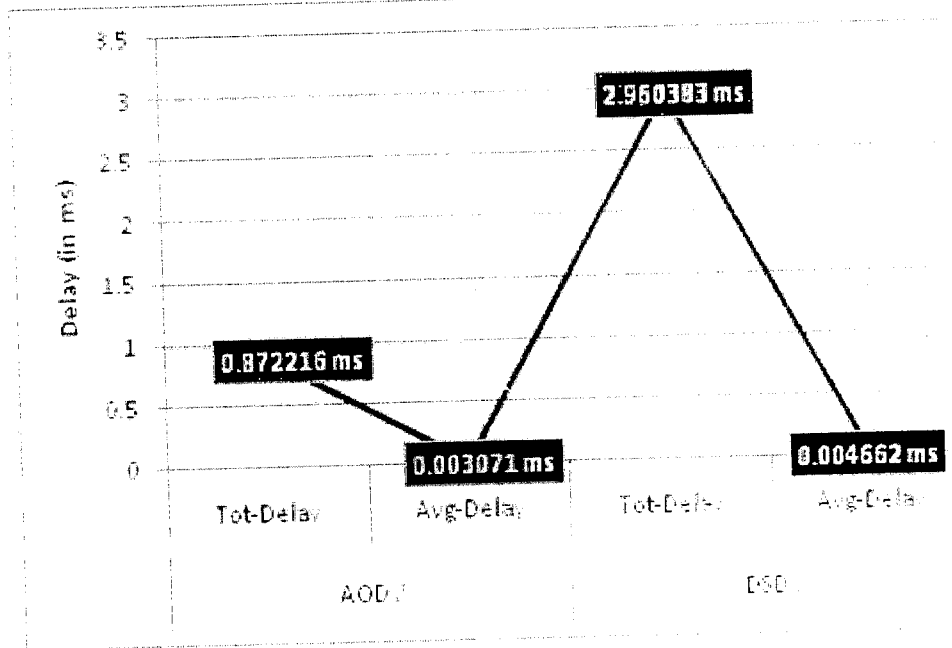


Figure 5.1[d] Comparison of Delay between AODV & DSDV Routing protocol.

5.2 ANALYSIS OF AODV & DSDV

Network Simulator (NS2) is C++ based discrete event simulator equally good for simulating both the wired and wireless networks. NS2 is one the favourites of the researchers for analyzing protocols since its support for wide variety of domains and its accuracy of results between the real and simulated environment. Our selected

Protocols AODV and DSDV can be best simulated using NS2. Two parameters are considered for routing protocols, which are Delay and Packet loss. A simulation topology has been created as shown in table I of simulation parameters. The following parameters are considered for simulation:

5.3 DELAY

“Delay can be defined as the difference between the packet generation time at the source, to the packet arrival time at the destination”.

5.4 PACKET LOSS

“Packet Loss can be defined as the packets sent by the source and the packets dropped (loss) before receiving by the base station (sink)”.

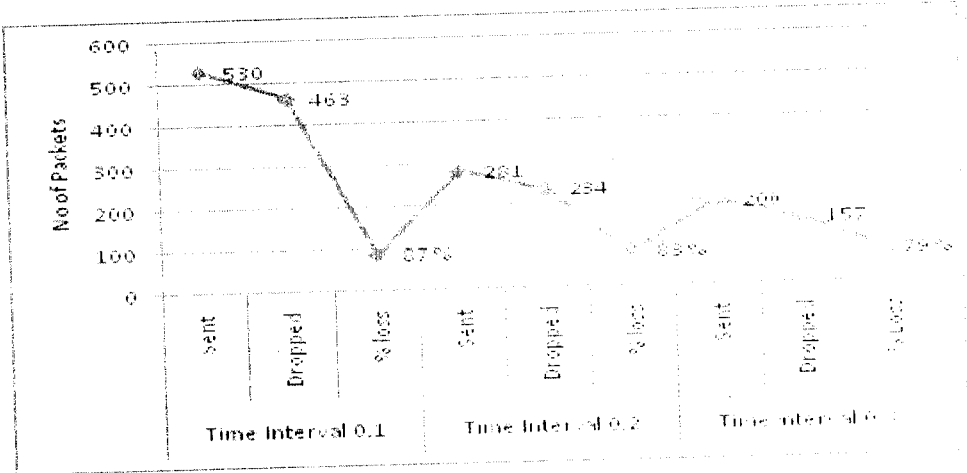


Figure 5.4[a] Packet loss of AODV

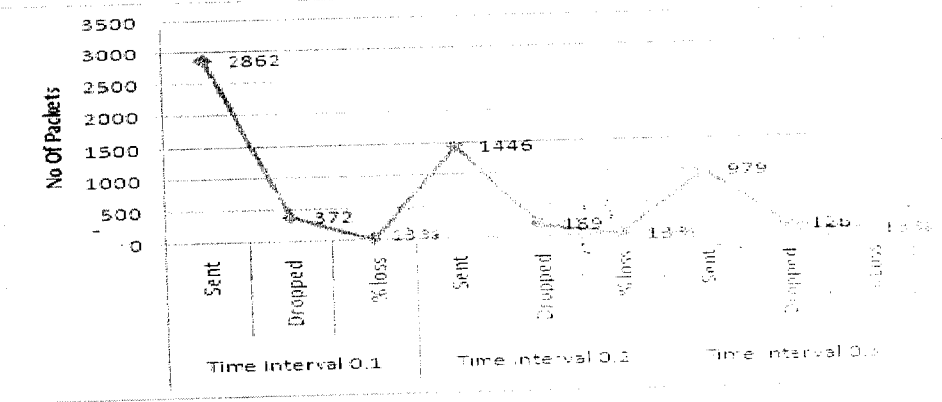


Figure 5.4[b] Packet loss of DSDV

5.5 RANDOM NUMBER GENERATION

The RNG class contains an implementation of the combined multiple recursive generator MRG32k3a proposed by L'Ecuyer [L'E99]. The C++ code was adapted from [LSCK01]. The MRG32k3a generator provides 1.8×10^{19} independent streams of random numbers, each of which consists of 2.3×10^{15} substreams. Each substream has a period (i.e., the number of random numbers before overlap) of 7.6×10^{22} . The period of the entire generator is 3.1×10^{57} . Figure 1 provides a graphical idea of how the streams and substreams $_t$ together.

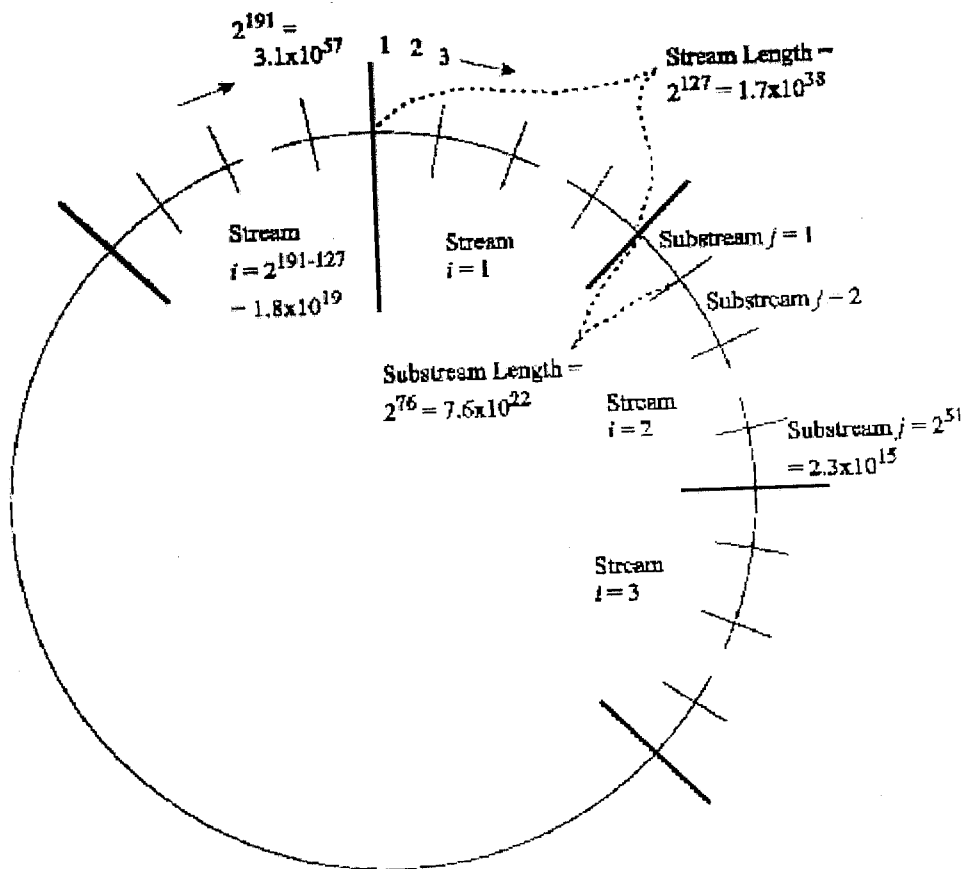


Figure 5.5[a] Overall arrangement of streams and substreams. [LSCK01]

A default RNG (defaultRNG), created at simulator initialization time, is provided. If multiple random variables are used in a simulation, each random variable should use a separate RNG object. When a new RNG object is created, it is automatically seeded to the beginning of the next independent stream of random numbers. Used in this manner, the implementation allows for a maximum of 1.8×10^{19} random variables.

Often, multiple independent replications of a simulation are needed (i.e., to perform statistical analysis given multiple runs with fixed parameters). For each replication, a different substream should be used to ensure that the random number streams are independent. (This process is given as an OTcl example later.) This implementation allows for a maximum of 2.3×10^{15} independent replications. Each random variable in a single replication can produce up to 7.6×10^{22} random numbers before overlapping.

Note: Only the most common functions are described here. For more information, see [LSCK01] and the source code found in `tools/rng.h` and `tools/rng.cc`. For a comparison of this RNG to the more common LCG16807 RNG (and why LCG16807 is not a good RNG), see [L'E01].

5.6 SEEDING THE RNG

Due to the nature of the RNG and its implementation, it is not necessary to set a seed (the default is 12345). If you wish to change the seed, functions are available. You should only set the seed of the default RNG. Any other RNGs you create are automatically seeded such that they produce independent streams. The range of valid seeds is 1 to MAXINT.

To get non-deterministic behaviour, set the seed of the default RNG to 0. This will set the seed based on the current time of day and a counter. This method should not be used to set seeds for independent replications. There is no guarantee that the streams produced by two random seeds will not overlap. The only way to guarantee that two streams do not overlap is to use the substream capability provided by the RNG implementation.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 THE SIMULATION ON NS-2

The MAMPR and AODV protocols are simulated in NS-2 network simulator, a discrete event simulator that can model and simulate multi-hop wireless ad hoc networks. The operating system used is Fedora Linux 9.0. The Distribution Coordination Function of the IEEE standard 802.11 for wireless LANs is used as the MAC layer. The radio model used is one similar to Lucent's Wave LAN with nominal bit-rate of 2 Mbps. The simulation is carried out with Constant Bit Rate traffic. Each sender sends data packets of 128 bytes long. The simulation used a send buffer of 50 data packets, containing the data packets waiting for a route. Packets sent by routing layer are queued at the interface queue till MAC layer can transmit them. The size of the interface queue used is 50 packets long. The other considerations made for the simulation environment and protocol settings are shown in table 6.1

6.2 RESULTS AND ANALYSIS USING NAM

NAM stands for Network Animator. This tool animates the network elements as described in the tcl script. A complete visualization is available to the user which depicts the networking concepts in the project. The animator takes the tcl file as the input and creates a nam and a trace file as outputs. NAM consists of tools for editing the network topology, navigation bar and a step size controller for time. With all these tools it is possible to vividly view how actually the project works with finer resolution in time. The tools for editing include zoomer and controls for interfacing. A status bar at the bottom of the animator indicates the current status of the network elements.

Table 6.1. Simulation settings and parameters of trust base technique

No. of Nodes	51 nodes
Area Size	1000 X 1000
Mac	802.11
Radio Range	100m
Simulation Time	150 sec
Packet Size	50

NS-2 can simulate the following:

- 1. Topology:** Wired, wireless
- 2. Scheduling Algorithms:** RED, Drop Tail,
- 3. Transport Protocols:** TCP, UDP
- 4. Routing:** Static and dynamic routing
- 5. Application:** FTP, HTTP, Telnet, Traffic generators

6.3 PHYSICAL/MAC/NETWORK LAYERS SPECIFICATIONS

Channel type	Wireless Channel
Antenna type	Omni Antenna
Transmission range	250 meters
Radio Propagation model	Ricean/Rayleigh
MAC protocol	Mac 802_11
Queue type	Priority Queue
Max queue length	50
Ad hoc Routing protocol	DSDV

6.4 PERFORMANCE METRICS

Nodes actual behaviors comply with the Bernoulli trial, which means that the probability that a node acts good is predetermined. If a node acts well for less than 40 percent of the interactions, it is considered as a misbehaving node.

The default percentage of misbehaving nodes in the network is 20 percent. Each node monitors other nodes in the same region and records the number of good or bad activities IN α toward each node. All nodes remain static and build up reputations in their home region in the initialization period.

The performance is evaluated according to the following metrics.

- **False positive**

In case of network failure, nodes may be falsely accused of misbehavior. The false positive should be kept low. The false positive can be detected at all types of scheme. False positive shows that overhearing the nodes produce.

- **Detection efficiency**

The detection efficiency of the activity-based and combined overhearing even increases at time. This is due to the larger number of routing protocol packets that circulate in the network. This enables the activity detector to predict more precisely whether another host is still in communication range.

- **Delay constraint**

The delay constraint is averaged over all surviving data packets from the sources to the destinations.

- **Untrust**

Untrust can be defined as the firm disbelief in the competence of an node to act undependably, insecurely, and unreliably within a specified context.

- **Throughput**

The throughput is averaged that the number of packet received without loss at the output. It is the measurement of how fast the packets that carrying the information can pass through a point.

6.5 PERFORMANCE RESULTS

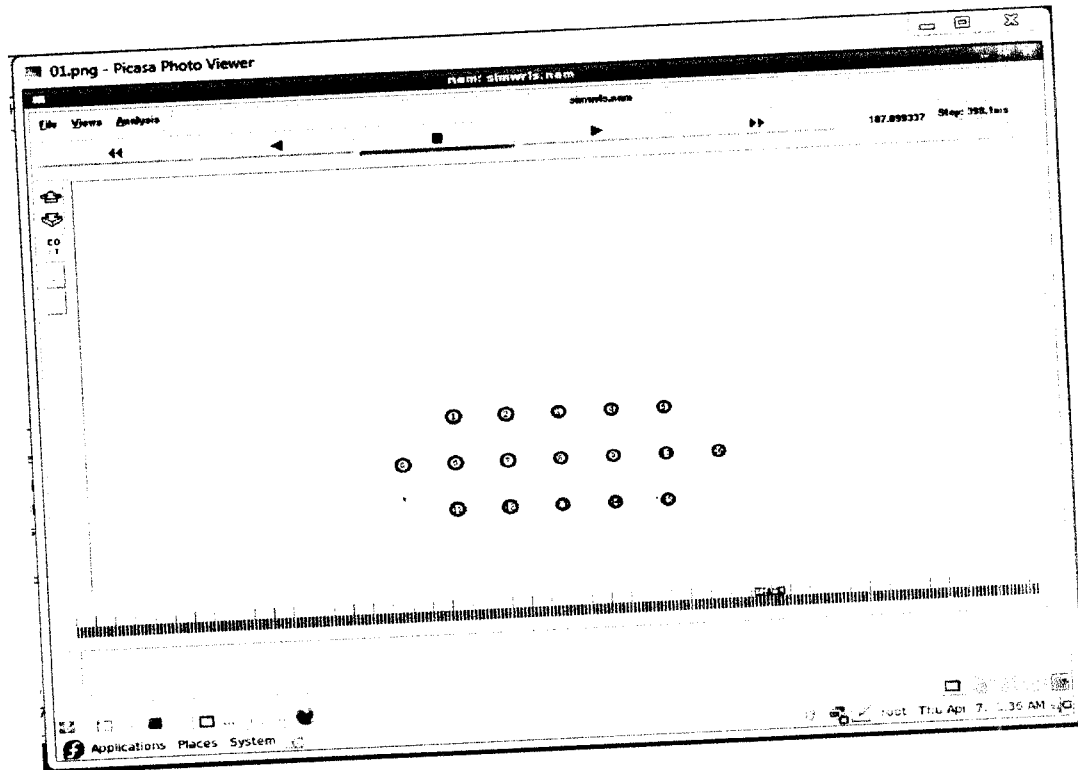


Figure 6.5[a] 17 nodes simulation

Fig 6.5[a] shows the simulated output with 17 nodes using AES. Here node '0' is treated as source node, it encrypts the data packets using AES algorithm. The node '16' is treated as destination node, finally it receives all the packets and decrypts using AES algorithm and retrieves the original data transmitted from the source node.

Fig 6.5[b] shows the simulated output with 30 nodes using AES. Here node '0' is treated as source node and node '29' as destination. Fig 6.5[c] shows the simulated output with 51 nodes using AES. Here node '0' is treated as source node and node '51' treated as destination.

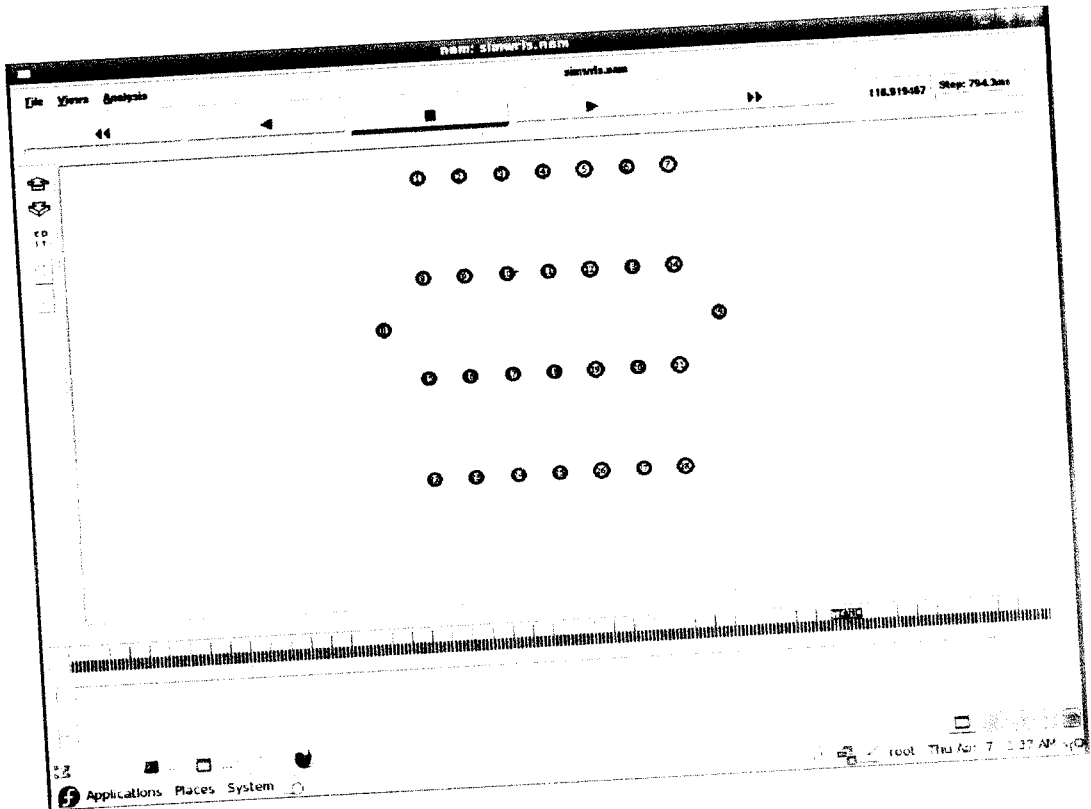


Figure 6.5 [b] 30 nodes simulation

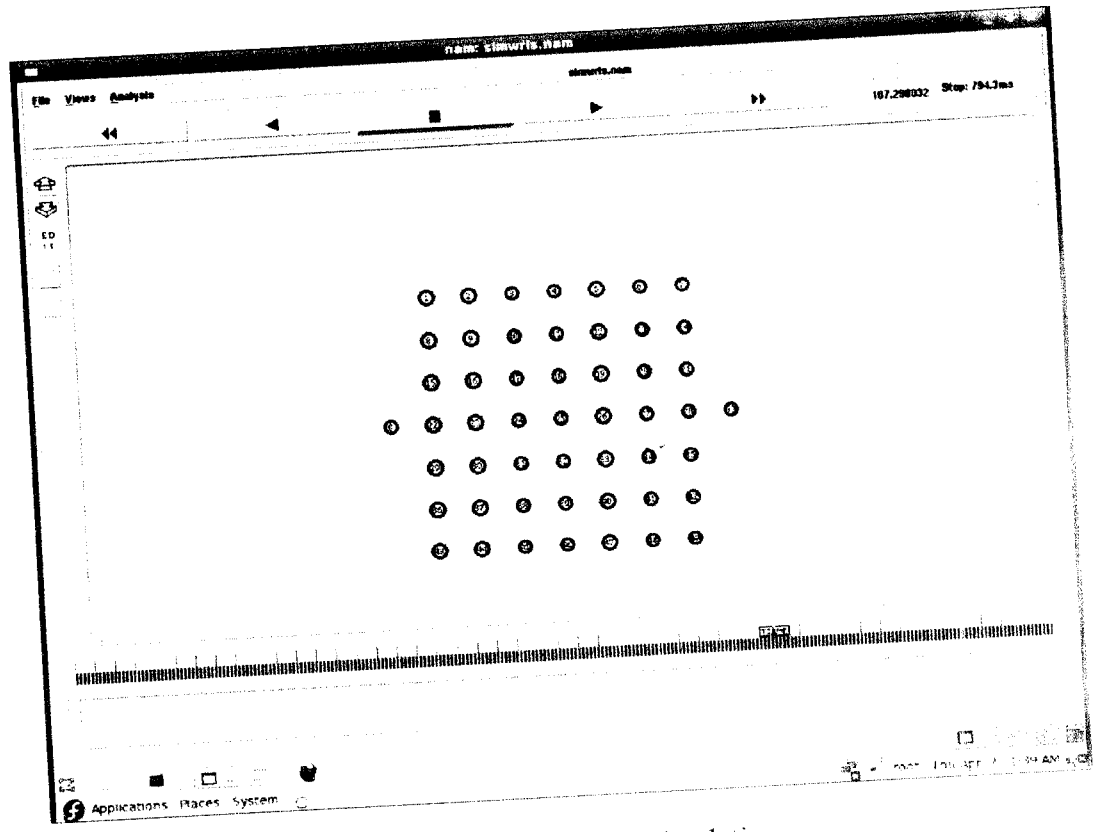


Figure 6.5[c] 51 nodes simulation

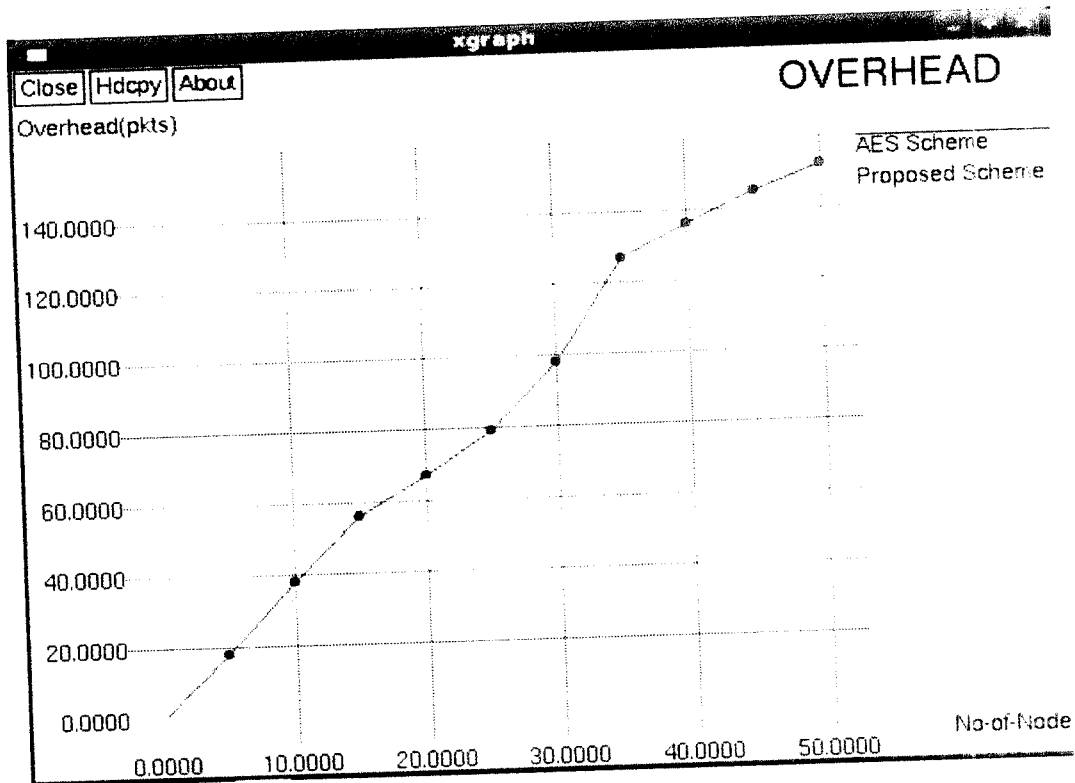


Figure 6.5[d] Overhead

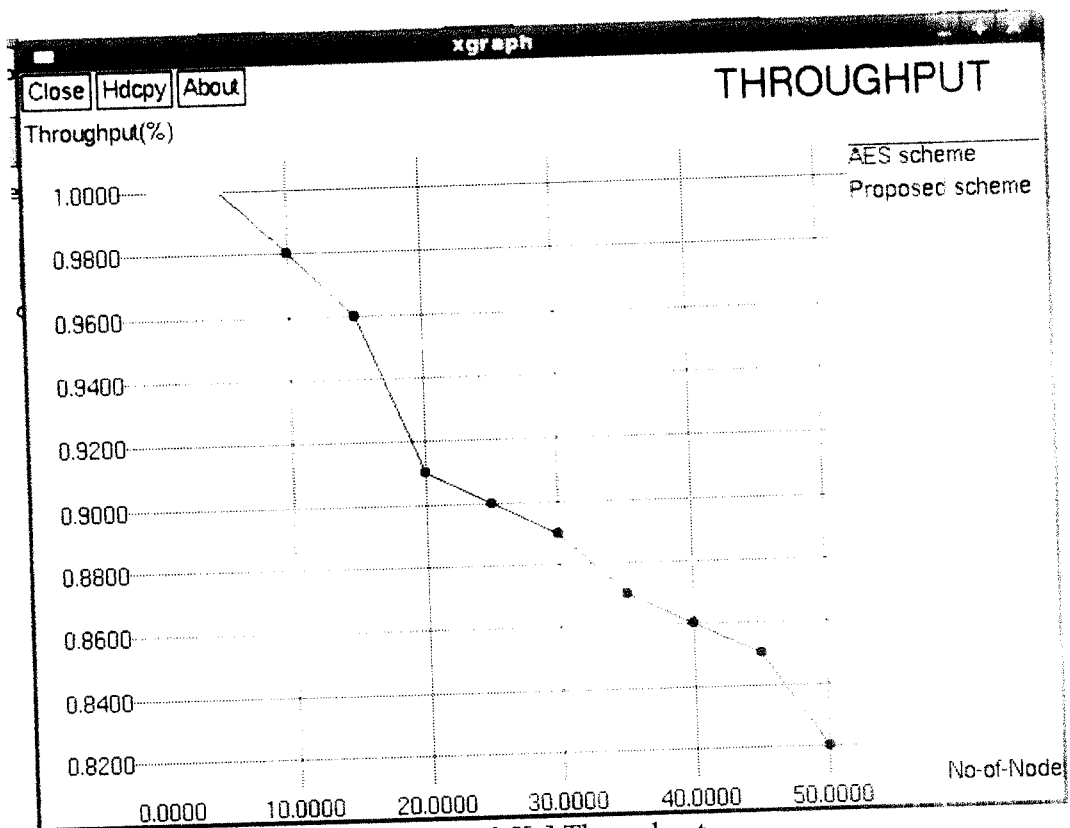


Figure 6.5[e] Throughput

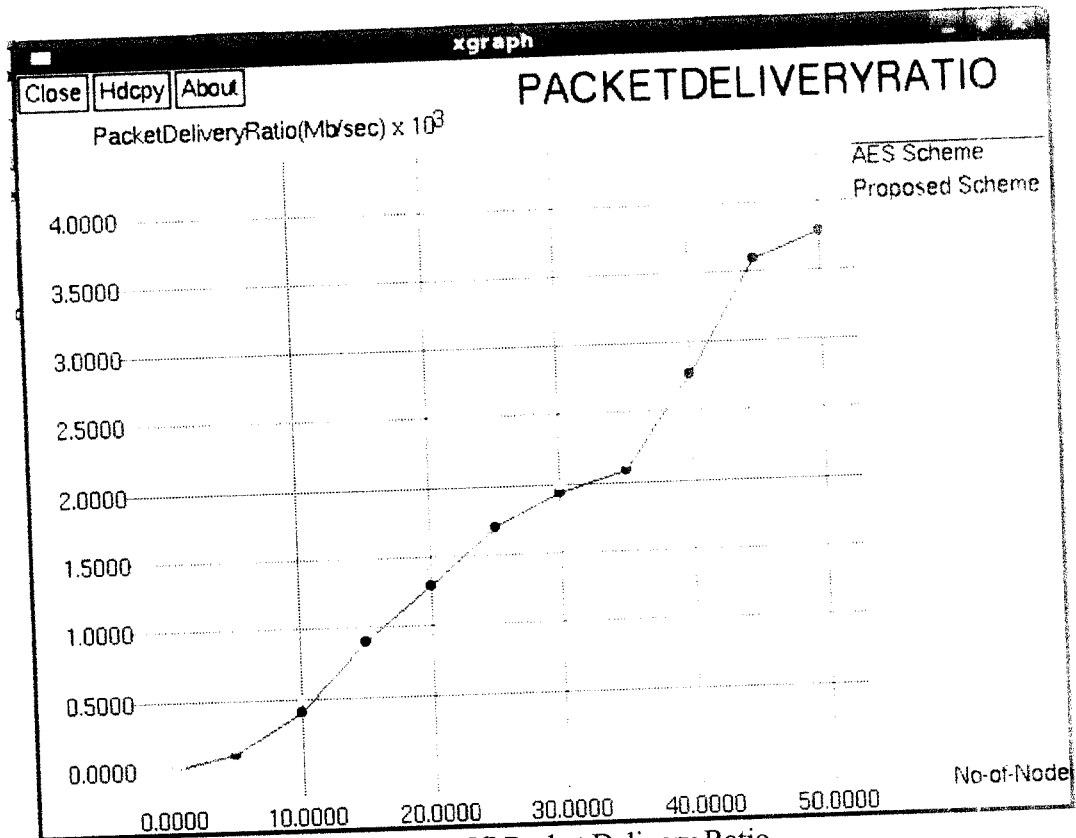


Figure 6.5[f] Packet Delivery Ratio

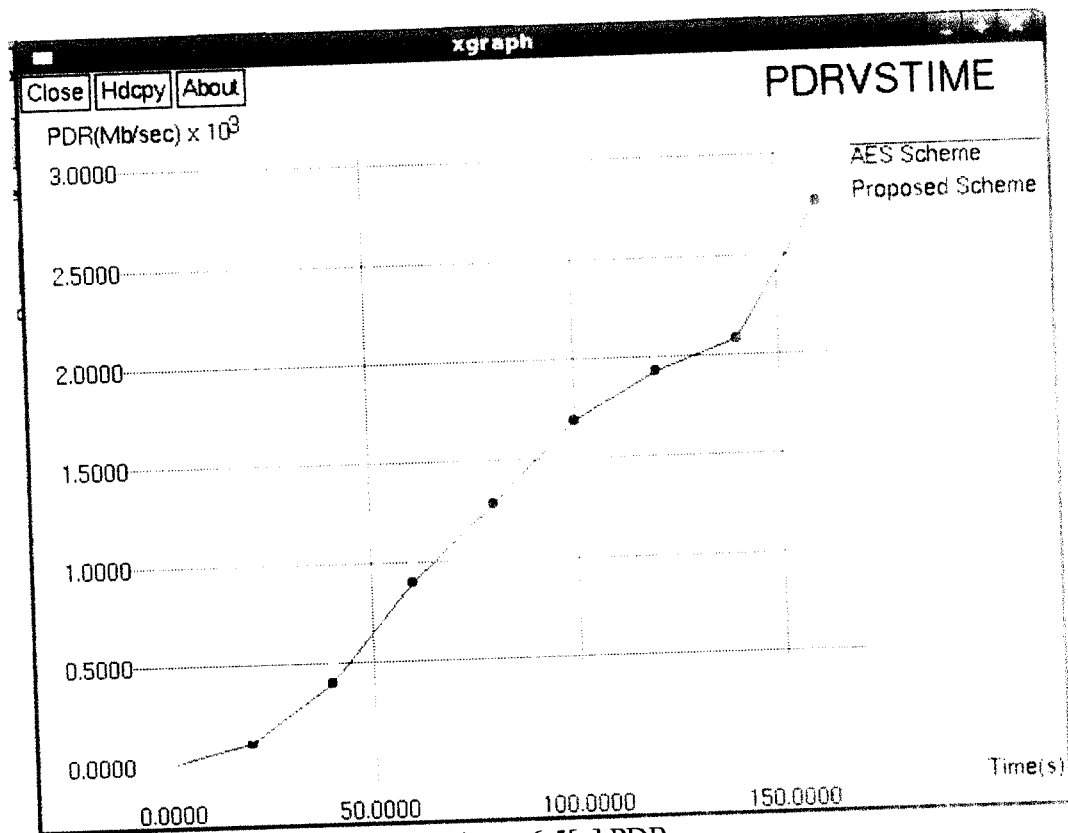


Figure 6.5[g] PDR

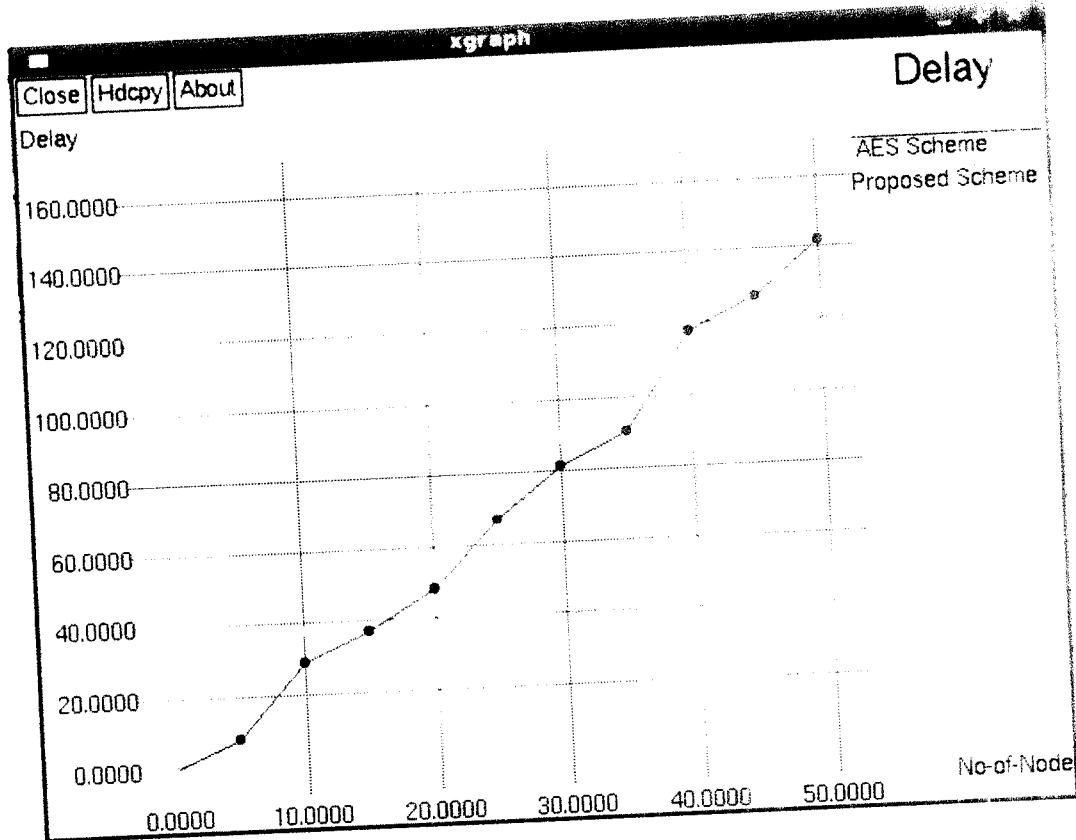


Figure 6.5[h] Delay

6.6 INFERENCE

From Fig.6.5[d] we observe that the proposed scheme has less overhead compared to the existing one. From Fig.6.5[e] we infer that the throughput is comparatively high in the proposed one than the existing one with the increase in number of nodes. Fig.6.5[f&g] shows the Packet Delivery Ratio(PDR) is comparatively better in the proposed scheme than in the existing one. Fig.6.5[h] shows the delay is slightly high in the proposed scheme than the existing one.

CHAPTER 7

CONCLUSION

Based upon the thorough theoretical analysis it can be concluded that the proposed scheme is efficient both in terms of security and efficiency. It provides a high degree of security against brute force attack which is the most common type of cryptanalytic attack. Also it reduces the overall energy consumption in the wireless sensor network by making the nodes secure enough to use a single (or small pool of) network-wide shared key(s) by ensuring that the nodes cannot be compromised by a remote attacker by simply decrypting the vital information present in a node using brute force attack. Thus it greatly reduces the communication required in the complex keying schemes used in key-management techniques and as 80% of the overall energy of a node is used up in communication, this scheme can help in prolonging the longevity of the nodes in a sensor network.

The proposed Dynamic cryptographic algorithm scheme is highly secure than the existing AES scheme, because it performs the XOR operation twice in all the intermediate nodes.

CHAPTER 8

FUTURE WORK

The proposed scheme can be implemented in both wired and wireless sensor networks for better security. For simplicity, this scheme will also be tested with RSA, DES Symmetric Encryption Protocols.

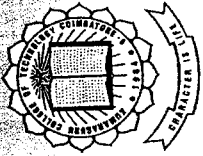
REFERENCES

- [1] Nivedita Mukherjee, "A Dynamic Cryptographic Algorithm To Provide Nodal Level Security In Wireless Sensor Network", International Conference on Innovative Computing and Communication- 2010.
- [2] Yang Xiao, Venkata Krishna Rayi, Bo Sun, Xiaojiang Du, Fei Hu e, Michael Galloway, "A Survey of Key Management Schemes in Wireless Sensor Networks".
- [3] A Perrig. Et al., "SPINS-security protocol for sensor networks", Proceedings of ACM MOBICOM – 2001.
- [4] D. Malan, M. Welsh, M.D. Smith, "A publickey infrastructure for key distribution in TinyOS based on elliptic curve cryptography", in Proceedings of 1st IEEE International Conference Communications and Networks (SECON), Santa Clara, CA - October 2004.
- [5] A.S. Wander, N. Gura, H. Eberle et al., "Energy analysis of publickey cryptography for wireless sensor networks", in: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications (PERCOM) - 2005.
- [6] M. Eltoweissy, M. Moharrum, R. Mukkamala, "Dynamic key management in sensor networks", IEEE Communications Magazine 44 (4) (2006), pp. 122130.
- [7] P. Traynor, H. Choi, G. Cao, S. Zhu, T. Porta, "Establishing pairwise keys in heterogeneous sensor networks", in: Proceedings of IEEE INFOCOM - 2006.
- [8] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, "Strong Authentication for RFID Systems Using the AES Algorithm", CHES'04, LNCS 3156, pp. 357{370,- Springer-Verlag - 2004.
- [9] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen, "AES Implementation on a Grain of Sand", IEE Proceedings on Information Security, Volume 152, Issue 1, pp. 13{20 - 2005.

[10] Deva Seetharam and Sokwoo Rhee, Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN04), pp. 0104.

[11] William Stallings, "Cryptography and Network Security Principles and Practice", 4th Edition , pp66,table22

[12] Marc Greis "Tutorial for the Network Simulator 'ns'.



KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology, Coimbatore)


COIMBATORE - 641049, TAMIL NADU, INDIA.




CERTIFICATE CITEL 2011

This is to certify that Mr./Ms. M. MUTHUKUMAR, MEC COMMUNICATION SYSTEMS
KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE has attended / presented a paper
titled A Dynamic Cryptographic algorithm for Wireless Sensor Networks in

the 3rd National Conference on **COMMUNICATION, INFORMATION AND TELEMATICS**
(CITEL 2011) on 3rd & 4th March 2011, organized by the Department of Electronics and
Communication Engineering, Kumaraguru College of Technology, Coimbatore.


Dr. Rajeswari Mariappan
HOD - ECE


Dr. S. Ramachandran
Principal


Dr. J. Shanmugam
Director