SEA: A FRAMEWORK TO ENSURE SECURITY, EFFICIENCY, AVAILABILITY OF DYNAMIC WIRELESS SENSOR NETWORK

by

Sumanth Gundappagari

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2020

I dedicate this thesis to my father, my greatest source of inspiration; to my mother, for her unconditional love and to my beloved sister.

నేను ఈ థీసిస్‌ను నా ప్రియమైన తల్లిదండ్రులకు మరియు నా చెల్లెలికి అంకితం చేస్తున్నాను.

Table of Contents

List of Tables

List of Figures

Abstract

Growing demand for the wireless sensor network has made it a hot research area and a new technology trend. Since sensor network technology can adapt to its environment, draw conclusions and learn from it, it makes more strategic use of the application environment in an efficient manner. It has now become widely available to deliver various advantages, from convenience, security, efficiency to time and money savings. Studies show that in the near future, wireless sensor networks will become more of an essential part of our lives than our actual personal computer. It is changing the face of the world altogether.

The wireless sensor network base station, which acts as a processing unit and gateway for all sensor data, is believed to be free from security threats. Unattended wireless sensor networks, however, should treat security as a primary concern, because it collects and processes sensitive data within networks. When the base station is compromised, the sensors connected to it are at risk and communications in the network are adversely affected. Significant security concerns surrounding these specifications include threats such as eavesdropping, data falsification, denial of service and disruption of physical nodes.

The proposed SEA (Security, Efficiency and Availability) framework is motivated from highly secure and scalable distributed ledger technology called Radix. The framework is designed to meet wireless sensor network security requirements using minimum replication and various key techniques. It promotes the use of public key cryptography among base stations to securely distribute keys, data, commitments and clocks. The proposed replication allows the network to continue functioning without data loss, and ensures the availability of the network. In addition, it can withstand vulnerabilities, such as man-in-the-middle and replay attacks. The proposed memory management scheme is validated in terms of memory storage and secure communication configuration. The security of message exchange in this framework is evaluated using Scyther protocol analyzer. The research findings exhibit that the solution proposed ensures security, efficiency, and availability of dynamic wireless sensor networks.

List of Abbreviations and Symbols Used

k	total number of neighbour base stations
mem^{β}_{local}	local memory of base station
mem^{β}_{remote}	remote memory of base station
r	replication factor
s_i	a sensor node
x	number of sensor nodes
ID_{β}	base station ID
ID_s	sensor node ID
β_i	a base station
δt_d	time of the response message received from the destination base station
δt_s	broadcast message sent time from a base station β_i
δt_{diff}	time difference between broadcast configuration request and response
$\oplus$	XOR operation
h	hash function
$h(location)$	hash of base station's location
lc	Logical Clock
mem_{β}	base station memory
mem_s	sensor node memory
mem^{β}_{free}	free memory of base station
$mem^{\beta}_{occupied}$	occupied memory of base station
n	number of base stations
$priv_{\beta_i}$	private key of base station β_i
$pub\beta_i$	public key of base station β_i
sk_{β_i}	base station's symmetric key
$sk_{\beta_i\beta_j}$	symmetric key used between base stations β_i, and β_j

AAP	Aerostat acoustic payload
ACE	Analytical and Computing Engines
AODV	Ad Hoc On-Demand Distance Vector
ASW	anti-submarine warfare
ATS	Acoustic threatening sound
BATM	Blockchain Authentication Trust Module
BL	blast localization
BS	Base Station
BTM	Blockchain Trust Model
CBM	Condition Based Monitoring
CCT	Configuration Change Time
CH	Cluster Head
CNP	Centroid of Nodes in the Partition
CPU	Central Processing Unit
CSV	Comma separated values
EARS	Early attack reaction sensor
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
GPS	Global positioning system
GSP	Geographic Sink Placement
HMV	Harmony Memory Vector
IDE	Integrated Development Environment

INS	Inertial Navigation System
IoT	Internet of Things
LEACH	Low Energy Adaptive Clustering Hierarchy
MAC	Media Access Control
MCM	missile canisters continuous monitoring
MERS	Multi-Error Report Simultaneously
MESS	Multiple Enhanced Specified-deployed Subsinks
MRMS	Multipath Routing with Multiple Sink nodes
NSA	Networked Sensors and Actuators
P2P	peer-to-peer
PDP	Provable Data possession
PoS	Proof of Stake
PoW	Proof of Work
PP	Perimeter protection
PRNG	Pseudo Random Number Generator
PRV	Pressure Relief Valve
PSO	Particle Swarm Optimization
R-IP-DS	Raw Information and Processed Data Storage
RCS	remote chemical sensor
SDL	Sniper detection and localization
SDT	Soldier detection and tracking
SF	Sensing Frequency
SHLM	Self-healing land mines
SLR	Systematic Literature Review

ST	Sensing Time
VDM	vapor detection with micro cantilever array sensors
WSN	Wireless Sensor Networks

Acknowledgements

Foremost, I express my sincere gratitude to my supervisor, Dr. Srinivas Sampalli, for encouraging and guiding me throughout my masters program. As a supervisor, he brought many remarkable inputs and encouraged me in my work presented in this thesis. He also involved me into different research projects, that gave me the opportunity to gain knowledge of great subjects and also make new friends. Your support helped me greatly in my success here. Thank you for always being so cool and understanding.

It was a wonderful opportunity and I am honored to be the first student under the guidance of Dr. Saurabh Dey. I thank you for supporting this thesis idea and giving such thoughtful feedback. I am deeply indebted to you for all your help in every aspect until the end of my research. You have been a perennial fountainhead of kindness, an inspiring guide who helped me thoroughly in improving my thesis quality and also in the creation of a fantastic looking document using Latex.

I would like to thank my committee members, Dr. Qiang Ye and Dr. Musfiq Rahman for accepting to be a part of the committee. I am grateful to have such experts in my defense.

I must thank my Mom (Swarna), Dad (Venkatesham) and my Sister (Sushma) for supporting me and helping me overcome all the stress. I would like to thank Lakshmi Kande who has always been encouraging, caring and supportive throughout my thesis and beyond. Dear best friend Ramya Kota, thank you for being there in my tough times. Last but not the least, I would like to thank P. Yaswanth, K. Manish, Ch. Chandu, B. Yashwanth, M. Deepak, B. Rahul, D. Akhil, B. Rohit, D. Santosh and all my other friends for your continuous support and motivation.

Chapter 1

Introduction

1.1 Introduction to Wireless Sensor Networks

Wireless communication advancements have allowed the development of Wireless Sensor Networks (WSNs) consisting of small devices that cooperate and gather information together. These tiny sensing devices are called sensor nodes consisting mainly of CPU (for processing information) , memory (for storing data), transceiver (for receiving and transmitting signals from one node to the other) and a battery [74]. It is a network (WSN) comprising large number of small sensor nodes that are used in a certain area to detect, monitor and measure several events through wireless communication. The traditional WSN has two main components: the sensor nodes and the base station. The main function of the sensor nodes is to sense, collect and send the required data where it is needed. The base station (BS) also called as sink node, is where the information collected, received and sent to the user via the internet or through a central hub [90].

1.1.1 Applications

These wireless sensor networks have received significant attention in the last decade, with their wide range of applications. The wireless sensor networks today are extensively used in commercial and industrial sectors including environmental surveillance, disaster anticipation, smart traffic management, health care, process control, ecosystem protection etc. [40] Some of the key implementation fields include:

Military Applications:

In military applications, the major concern generally extends from data collection to enemy tracking, monitoring of the battlefield or target classification [59, 40]. Different classes of military applications include self-healing land mines (SHLM), aerostat

acoustic payload (AAP) for transient detection, soldier detection and tracking (SDT) , low-cost acoustic sensors for littoral anti-submarine warfare (ASW), early attack reaction sensor (EARS), sniper detection and localization (SDL), time difference of arrival blast localization (BL) using a network of disposable sensors, perimeter protection (PP), chemical, biological and explosive vapor detection with micro cantilever array sensors (VDM), A low-cost remote chemical sensor for E-UAV platforms (RCS), novel optical sensor system for missile canisters continuous monitoring (MCM), inertial navigation system (INS) and Acoustic threatening sound recognition system (ATS) [27]. Usage of wireless sensor networks in highly vulnerable areas and hostile environments such as international borders and large military barracks need wireless sensor networks with multiple base stations to increase network lifetime and security [32]. The accuracy of data and the availability of information is very essential in these types of applications. The effect of data loss could be catastrophic.

Power System Applications:

A power system comprises of three important components, i.e. generation, transmission, and distribution. Power System applications include - Thermal Rating and Monitoring of Conductors, Smart Grid Sensor and Actor Application, Performance Monitoring of Electrical Distributed Systems, Automatic Remote Meter Reading Systems [53, 11, 52]. The overall surveillance and fault detection of these power grid systems is very crucial. In addition, wireless sensor deployment requires prompt and accurate transmission of observed information to ensure efficient monitoring of the power system [70].

Oil, Gas and Resources Industries:

Specific applications for Oil and Gas Remote Monitoring using the wireless sensor network technology are Pipeline Integrity Monitoring, Tank Level Monitoring, Equipment Condition Based Monitoring (CBM), Pipeline Pressure Relief Valve Monitoring (PRV), Refineries Pressure Relief Valve Monitoring (PRV), Wellhead Automation and Monitoring and several others[3].

Critical applications which include military applications, health care and robotic

wireless sensor networks can be good examples of a dynamic network to use proposed framework. Border protection/ surveillance where military tanks or vehicles are equipped with base stations moving from one place to another. Health care services also consider wireless sensor network technology to provide reliability in addition to enhanced mobility such as in hospitals where doctors/nurse carrying base stations can sense and keep track of patients data using their devices such as mobile phones. Many emerging fields such as robotic wireless sensor networks having mobile robots moving from one place to another assigned with specific set of tasks can be another significant application area.

1.1.2 Challenges

The advancement of dynamic wireless sensor networks and its usage in critical sectors make it susceptible to various challenges, such as security, efficiency and availability of data. The use of sensor nodes in an unsupervised area makes the network prone to a range of possible attacks. Furthermore, conventional security solutions are impractical due to inherent power and memory limits of sensor nodes[65]. Since sensor networks can communicate sensitive information and/or employed in adversarial unsecured conditions, security issues need to be addressed from the start of network design[82]. The short lifetime, smaller coverage and connectivity range, maintenance and reliability are some of the challenges that limit the usefulness of WSN. For this reason, energy efficiency is a major barrier to implementing sensor networks[10, 30]. Additionally, the larger the sensor network, the more intelligent and responsive we become, the more useful and informative our visualization becomes. However, as the network size increases, so does the related complexity and secure management of data[33].

Despite serious resource limitations of wireless sensor nodes, some of the current extremely important security requirements are:

Data Confidentiality:

A sensor network should not be able to leak sensor measurements; data stored within the sensor node may be highly sensitive particularly in the military application. The standard method for keeping sensitive information is to encrypt data with a private

key, which is completely secret for only intended recipients[82].

Data Integrity:

An attacker may be unable to obtain information by enforcing confidentiality. However, this does not mean that the data is protected. The integrity of the data ensures that no data obtained was manipulated during communication[82, 10].

Availability

The phrase "data availability" refers to the ability to guarantee that the data is always available to users even when breakdowns occur[65, 82].

Data Freshness

The freshness of the data indicates that the information collected is new and that there is no relaying of older duplicate messages[44, 65, 82].

1.2 Motivation and Objectives

Most recent WSN work focused on increasing lifespan, energy efficiency and reliability through enhancement of existing routing protocols[9], employing relay nodes deployment[29], implementing energy efficient clustering algorithms[68], wireless power transfer to sensor nodes[16], dynamic channel sensing and topology control mechanisms[72]. A lot of research has been done on the security of WSNs to this day in terms of increasing confidentiality, data integrity, authentication with the use of secure routing protocols[89, 34, 35], key management protocols[34, 42] etc. Nonetheless, very few research studies concentrate on availability of information, data immutability and efficient data storage. Some important applications, such as the deployment of wireless sensor networks in military services[27], hospitals[40], nuclear power plants[28], oil and gas industries[3], require continuous data availability[80] even in the event of an attack. In order to improve service availability few research works considered multiple base stations deployment. In all those researches little or no cooperation is found among the BSs. Ultimately, related data is lost from the base station and some areas of the network often go orphaned due to the BS compromisation. Data immutability

implies the integrity of data when it is stored in the BSs. Related researches have discussed the integrity of data while in transit but do not provide a comprehensive protection that guarantees data integrity at the BSs[39]. Previous researches that have focused on multiple base stations do not include the concept of data replication. Such replicated data helps to maintain the validity of the data and to ensure the availability of data when the base station is compromised.

This research proposes an efficient method for replicating data across base stations to tackle service availability, data immutability, data freshness at base stations. Public key encryption[81] is also used for the secure transfer of data between base stations without the need of key management protocols. Every base station or sink node maintains and partitions its data storage capacity into two logical sections. One for local storage and one for remote data storage. Local storage is the logical memory partition which includes data collected from the sensor nodes directly connected to the base station, while remote storage is the other logical memory partition used to store the copy of the local storage of other base stations. This thesis suggests a way to efficiently store the data logically and securely at the base stations. It also clears memory and increases storage capacity when necessary by sending information online to the server. This approach uses a variety of techniques such as replication, the use of logical clocks and the concept of commitments to achieve data availability, data integrity, prompt data synchronization and data freshness at base stations. With a certain amount of data duplication in the network, no data will be destroyed even in the case of failure or compromisation of the base station. This method ensures that a compromised base station does not result in data loss as the complete and total replicated information is present in the storage of another base station(s) and is subsequently sent to the server. The presence of black hole nodes and compromised base stations can eventually be identified and tracked through the utilization of periodic commitments.

1.3 Thesis Contribution

Much work has been performed so far on the security of WSNs. The emphasis of most of these research works is anonymity, integrity and authentication through the design of energy-efficient cryptography protocols. Most of these works, however, are

not very concerned about supporting the network with better service availability, data freshness and immutability. Service availability implies the availability of sensor data in base stations even when some nodes are affected in the network. Data freshness is the presence of recent data at base stations rather than outdated and obsolete data. Immutability is the integrity of base station data, which cannot be altered or changed. Many researchers assume that the base station is secure and they emphasize more on sensor node security protocols and the secure data transmission between the sensor node and base station. This thesis focuses primarily on security requirements such as data availability, data freshness and immutability of information. To satisfy all these requirements, the proposed framework adopts the concept of asymmetric key encryption, data distribution among a network of multiple base stations and logical clocks. Some of the ideas are powered by a new innovative distributed ledger technology called Radix, with a built-in consensus architecture which is highly scalable, secure and efficient. In addition, the proposed framework prevents security issues such as replay attacks, node compromise, etc.

Chapter 2

Background

2.1 Sensor Network Model

Wireless Sensor Network (WSN) is a self-configured network for physical or environmental monitoring of conditions such as temperature, sound, vibration and pressure. Sensor nodes collectively transfer the sensed data to a central location called base station or sink node where the data is gathered and is analysed. There are more than hundreds and thousands of sensor nodes in a traditional sensor network. A base station or a sink node is the central location which acts as a network-to-user interface. Information needed from the network can be obtained by inserting queries and collecting data from the sink. Most of the recent networks are bi-directional, which also allows for sensor activity monitoring and control. Military applications such as battlefield surveillance inspired the development of wireless sensor networks, that are nowadays used in various industrial and business applications such as industrial process management and surveillance, health monitoring, etc. [58]. The main components of traditional WSN are shown in the fig. 2.1.

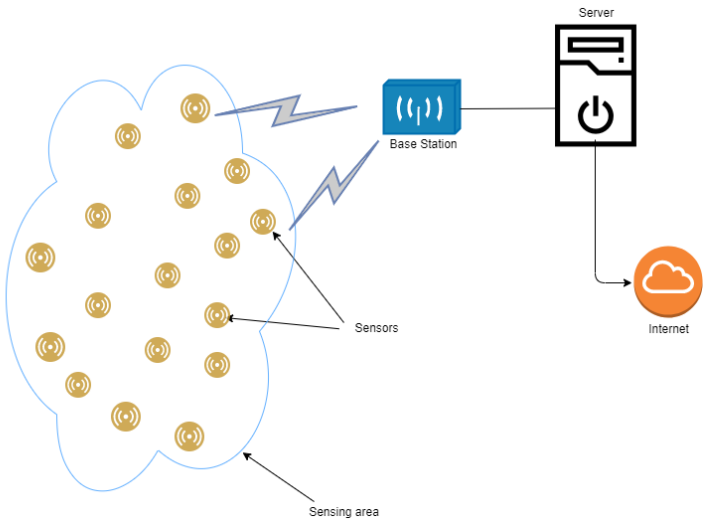

Figure 2.1: Traditional wireless sensor network model

Sensing area: The region in which the nodes are located is called as a sensor area.

Sensors: Sensor nodes are the core of the network and are responsible for data collection and redirection to the sink.

Base Station: A Base station is a node that collects, processes and stores data from other sensor nodes for a specific purpose. It is the central control point of the network which also establishes a bridge to other networks serving as an efficient data processing and storage center. Base station is also a point of access for the user to control the network.

For several motivating factors, recent approaches consider using multiple base stations. Multiple base stations not only increase network life, but are also capable of enhancing network security. The main focus of this research is to enhance the security of the wireless sensor network taking into account data availability integrity, confidentiality and data freshness. Fig. 2.2 shows a standard topology of multiple base stations deployed in a sensor network.

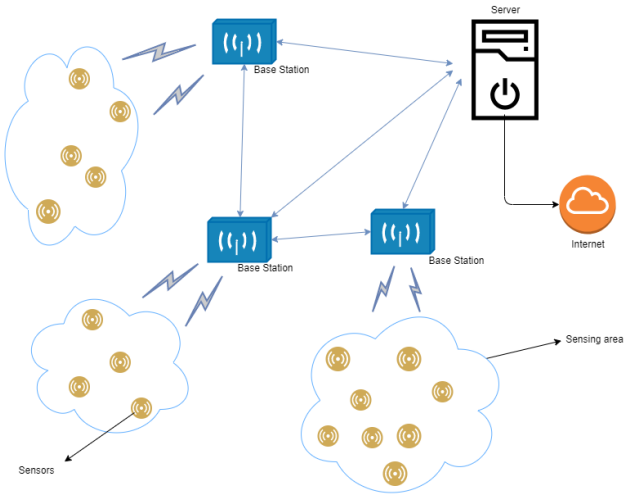

Figure 2.2: A standard topology of multiple base stations deployed in a sensor network

2.2 Sensor Network Hardware Technology and Communication Architecture

A wireless sensor node is a device consisting primarily of a microprocessor, a memory, a radio transceiver, a power source and one or more sensors attached to it (Fig. 2.3). Sensor nodes can also include a Global positioning system (GPS) to receive location information if required.

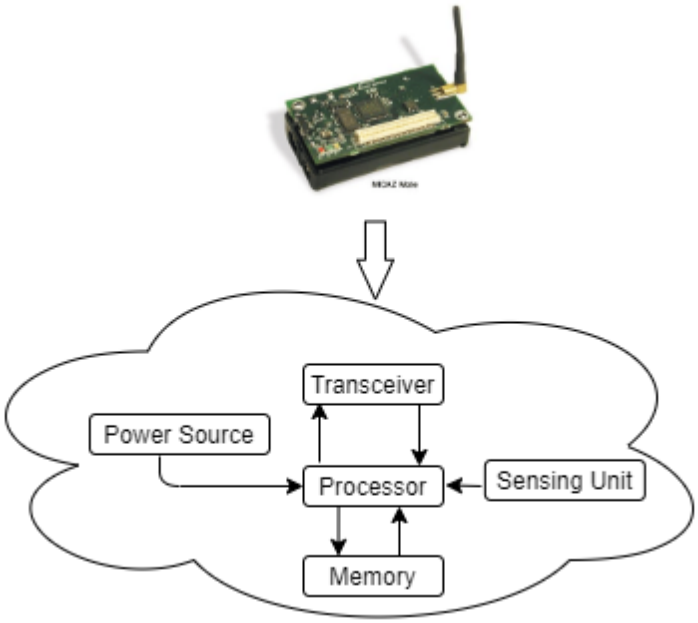

Figure 2.3: Sensor Node Architecture

The transceiver is used to communicate with other sensors and the base station via wireless links, while external parameters and data sensing is measured by the sensing unit. Additional elements such as power harvesting modules or actuators may also be mounted with sensors. To minimise the overall cost of the system, the price of nodes must be kept low, and a decent trade-off must be found between the amount of functionalities that the sensor offers and its price. The processing unit analyses sampled data and stores data on the memory unit. Sensor nodes usually are powered by batteries, but can also scavenge energy from their surroundings[14]. The potential of the network to auto-heal and to make adjustments itself to changing environmental conditions is another significant benefit when using WSNs for data data collection and reporting. Even if WSNs manage wireless connection they do not rely on costly

communications infrastructure. For a diverse variety of applications, WSNs utilize small and low cost embedded units and do not rely on existing infrastructure for operation.

Three types of communication patterns can be addressed in this type of network: broadcast, Group communication and Node to Node gossiping. Broadcasting is usually used by a base station (sink) to transfer certain information into all of the network's sensor nodes, such as network configurations, requesting sensor data, sending configuration updates, etc. Node to Node gossiping is a type of communications method in which a sensor node sends a message to other sensor node to relay the data back to the base station. Sensor node to base station communication and base station to base station communication also fall within this node to node gossiping category. Finally, group communication is the process where a set of sensors interact within themselves or with a base station.

2.3 Blockchain

Blockchain started when people wanted to timestamp digital documents so that it is not possible to backdate them or to tamper with them. This went by unused until it was adapted by Satoshi Nakamoto in 2008 to create the digital cryptocurrency Bitcoin [1]. Blockchain is a distributed ledger that maintains an immutable continuous connected list of data in the form of blocks [77]. It has an interesting property that the data recorded in blockchain is difficult to change. The term 'distributed ledger' represents a ledger system that contains more than one copy of the same ledger spread throughout the network. Distributed ledger across the network helps to minimize the risk and potential negative impact of data loss, data corruption or fraudulent alterations. If there are several replicas in the network, loss or corruption of a single copy will not affect the network[63]. Each block contains some data, nonce, the hash of the block and the hash of the previous block as shown in fig. 2.4.

The data that is stored inside the block depends on the type of blockchain, for example bitcoin blockchain contains the data related to transactions in bitcoins from one to another. Hash present inside the block is made using some hashing algorithm which always produces the same hash given the same set of input. This is used to identify that block and all of its contents, and is always unique just as a fingerprint

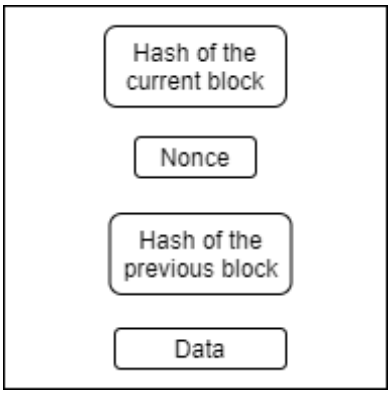

Figure 2.4: Blockchain Block example

for humans. Once a block is created its hash is calculated, even minor changes in the block can lead to a complete different hash which makes it useful to detect changes and manipulations in the block. The third elements inside this block is the hash of the previous block, this is used to create and join a chain of blocks forming a blockchain. Nonce is used to solve the hashing difficulty problem in bitcoin blockchain and to connect a linear sequence of blocks together. Fig. 2.5 shows how 3 blocks are connected to each other to form a chain of blocks.

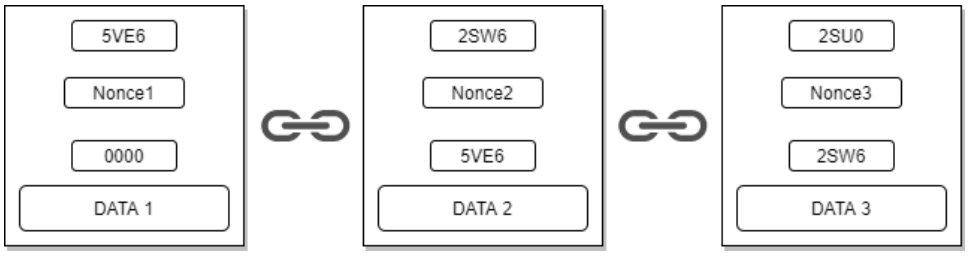

Figure 2.5: Example of Blockchain - chain of blocks

The previous block hash in the first block is 0000 as it is the first block and no other block is before it, also called as genesis block. The next block has the previous hash value of the first block i.e., 5VE6 and so on. If a blocks data is tampered in between, it causes the hash of the block to change as well. in turn making the following blocks invalid as they no longer store the correct hash. Nonce help the blockchain to secure the data and restrict others to create a new chain easily. Thus the security of bitcoin blockchain mainly comes from its creative use of hashing, nonce mechanism and the data being distributed. This approach creates consensus among all the nodes in the network. The blockchain protocol helps to make decisions to

agree about which blocks are valid and which are not.

The ledger is replicated across a peer-to-peer (P2P) platform among several users or devices in order to avoid centralization. When a node joins the blockchain, it gets full copy of the data. The blockchain network has users, miners and nodes. Three primary tasks that are performed by nodes within the blockchain network are receiving transactions, distributing or relaying transactions; updating blockchain databases with the current valid transaction records [15]. A blockchain node is a computer or a decentralized unit, which operates blockchain applications and stores the data connected to a blockchain network [88].

Current blockchain networks are categorized into three different forms; public (permission less), private (permissioned) and consortium or hybrid blockchains [85]. The public decentralized blockchain is an open network that is accessible to anyone with access to the Internet. Such blockchains have no centralized authority and often employ powerful consensus methods which need a huge amount of power to deter malicious users from interfering in the network [78]. Private blockchains have problems, especially when companies want to work together, as it is impossible for one another to read or access each others database. In that case, building a permissioned blockchain helps to share data between the two companies and to track the data transferring across different ledgers. Consensus algorithms and implementations in private or permissioned blockchains are not as powerful as public blockchains and do not need large amounts of power, however access to the ledger database is limited to authorized users. The owner of the private blockchain has complete access to add or remove authorized users. Identifying individuals and permitting them to access the ledger is required in private blockchain whereas public blockchain users are completely anonymous. Consensus algorithm is determined by the participants engaged in the private blockchain, and those who created this blockchain [88]. Between private and public blockchains are the hybrid blockchains in which a group of approved individuals or companies decide to share decision-making equally among themselves. The consortium blockchains are most often related to corporate or industrial use, where a group of companies interact with blockchain technology to support their businesses [76]. Blockchains are also constantly evolving, one of the most recent developments is the creation of smart contracts. These contracts are simple programs stored in the

blockchain network and used to execute transactions when some conditions are met. This technology is now used for other things like transferring of assets and so on [18].

As the name suggests, blockchain technology is a chain of connected blocks that comprises a small set of transactions or messages in each block. These blocks are linked with each other through the use of a cryptographic fingerprint called hash. This linear sequential order of the hashes in different blocks is referred to as chain. The transactions in blockchain has information about the message being added to the ledger, source address and destination address of the message, while the block has a group of these transactions along with the hash of that block and random number called nonce which is used to connect another block to it. A transaction in a blockchain is successful when it is sent to the blockchain node, verified using the signature of the source address, stored in a block and added to the blockchain after numerous trials of various nonce numbers.

2.4 Radix Distributed ledger Technology

Radix DLT is a new and unique platform that is highly scalable and easy to build on, unlike Bitcoin or Ethereum blockchain technology. The design is entirely different from blockchain and can be used by millions of users at the same time without compromising speed or centralization problem. The system is asynchronous and byzantine fault tolerant of errors, ensuring it can identify and avoid fraudulent transactions present in the network. Radix is a very reliable and highly scalable architecture primarily used for storing and accessing immutable data with a decentralised logic design. Moreover, to achieve overall system security, there is no need for large quantities of computing power (PoW) or huge amount of capital (PoS). This Decentralized Ledger Technology provides a revolutionary platform for digital data storage and verification of the stored data; it is also cryptographically robust, very hard to hack, and highly adaptable. Radix uses logical clocks, public key encryption and a hashing mechanism to accomplish all of these requirements. Even with 99 percent of malicious nodes on the network, Radix is able to identify and differentiate false data to reach consensus [69, 12]

The Radix Ledger has three main elements:

- A group of connected nodes

- A complete database spread across all the nodes
- An algorithm for generating secure cryptographical record of transactions

In this architecture, any event or transaction is represented by an object called Atom. Atoms are of two types, Payload atoms and Transfer atoms. An example of a payload atom is some information sent as a message where as Transfer Atoms are used to transfer the ownership of atoms.

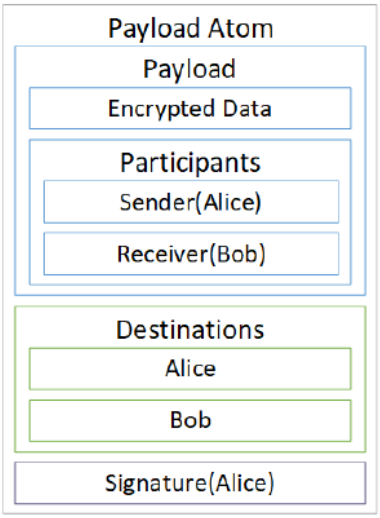

Figure 2.6: Example of payload atom [36, 69]

Fig. 2.6 is an example of payload atom containing data in an encrypted form along with source address and Destination addresses. Users can generate and send payload atoms to the network via any node to which they are connected. The Radix consensus algorithm (called as Tempo) then processes this transaction by checking if it is valid and then adds it to the ledger database. This information of submitting event to validate it by the nodes is called as a ledger event. Thus, an event is called a ledger event as it changes the state of the ledger. The current node after validation of the event, sends the information of this update to other neighboring nodes via broadcasting using gossip protocol. about the addition of new event, this is termed as protocol event. These two different events help Radix to achieve consensus with the use of Asynchronous Byzantine Fault tolerance for Ledger Events and Asynchronous Byzantine Fault detection for Protocol events. Tempo is very collaborative and fast as nodes communicate each other, move events to each other and ensure that all nodes

in the network update their ledger to eventually obtain consensus among the whole network [36, 12].

2.5 Elliptic Curve Cryptography

Cryptography is the study of secure communication methodologies in the presence of foreign entities called adversaries. It involves the development and analysis of the protocols that prevent private or secret messages to be read by foreign entities or unauthorized individuals. Cryptography plays a major role in various areas of information security, such as data confidentiality, data integrity, authentication and non-repudiation. Data encryption is the transformation of information from readable state to non-sensical data. Two different kinds of cryptography techniques exist, symmetric or private key cryptography and asymmetric or public key cryptography. Transmission of data encryption messages was by symmetric key until the end of 1970s. That indicates that a person with sufficient information to encrypt messages can also decrypt those encrypted messages. Nevertheless, new cryptography needs grew with the increased technological advances of economic life [46]. Public-key encryption is not previously developed mainly because it was not really needed until recently due to the computerization of economic life [54, 46].

Three main fundamental methods that are are available for public key cryptography systems are

- The RSA Algorithm [62]
- Diffie-Hellman Algorithm [24]
- Elliptic Curve Cryptography [87]

The most popular and widely used public key cryptography implementation was by Ronald Rivest, Adi Shamir and Leonard Adleman called as RSA Algorithm. It is used in various applications and is primarily implemented in digital signatures, key exchanges and data encryption.

Diffie and Hellman developed their own algorithm after the RSA algorithm was released to the public. Diffie-Hellman Algorithm is used only for the key exchange

mechanism, not for digital signatures or data encryption, as opposed to RSA Algorithm.

Elliptic Curve Cryptography (ECC) is a public key cryptography algorithm based upon elliptic curves. This algorithm is not vulnerable to those algorithms that make solving discrete logarithm problem easier. Elliptic Curve cryptosystem is invented in 1985 by Neal Koblitz and Victor Miller[41]. It is one of the most efficient and commonly used algorithms today. RSA and Diffie-Hellman were revolutionary when they were first released mainly due to the introduction of a new public key cryptographic scheme rather than symmetric key cryptography which removes the need to share secret keys. However, with the introduction of Elliptic curve cryptography, smaller ECC keys are sufficiently good to achieve the same security level as larger keys in other algorithms, such as RSA. Interesting property of elliptic curves is that, a non-vertical line drawn through any two points on an elliptic curve A and B will intersect the curve in at most one other point C. An example of an elliptic curve is shown in the fig. 2.7.

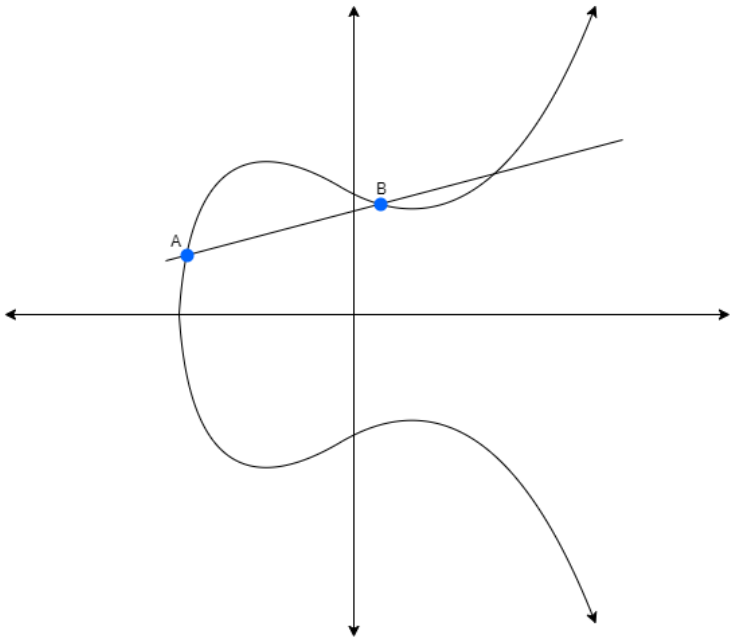

Figure 2.7: Example of Elliptic Curve

If we flip this new point C' over the x-axis, we get C. And if we continue to do this operation again and again for n number of times, we get randomly new n

number of points. This method of getting new points is called as dot operation. This operation of getting new random points is similar to that of exponential problem, even if we know a point on the curve, it is really impossible to figure the n (number of dot operations performed) for which that point is reached. Similar to Diffie-hellman, where $(G^n)^c = (G^c)^n$, the dot operation performs the same calculation in elliptic curve cryptography which is c dot $(nG) = n$ dot (cG).

An Elliptic curve E is a curve over finite space F which is not equal to 2 and 3 and is satisfying the equation

$$Y^2 = x^3 + ax + b[46]$$

In ECC an elliptical curve is a plane curve on a finite space comprised of points that follow the above equation where a, b belongs to the finite space F with pn elements, , where p is a prime number and a, b satisfying $4a^3 + 27b^20$ [50].

Using this concept of elliptic curve, it is mathematically much more harder to solve the elliptic curve discrete logarithm problem compared to RSA discrete logarithm problem. This makes it is much more efficient as it can get away with shorter key sizes, which means less computation when we are calculating. Thus it is much easier and helps better to compute faster as shorter keys mean smaller calculations and less data sent from server to the client. The standard ECC key of 256 bits equals a 3072-bit RSA key and is 10,000 times more powerful than an RSA key of 2048 bits. Additionally, the ECC requires minimal computational power (CPU) and storage, which allows for significantly higher response rates and performance on Web servers when used [2].

2.6 Logical Clocks and Hashing

Logical clocks is an algorithm created by Leslie Lamport in 1978. This is mainly used to determine the ordering of events in a distributed network. This algorithm helps to perfectly synchronize different nodes with the use of happened before relation. The main problem of the distributed computer system is the synchronization of events happening in the network. As the distributed system lacks a global clock and all the events in the system are unordered due to different timestamp in different device. This is called as the clock synchronization problem. Leslie Lamport have solved this

problem by the use of logical clocks for every process. It provides a partial ordering of events with the use of a counter which increments by a predefined value for every event encountered by the system. This partial ordering of events happens with each process P_i having a clock C_i. The time stamp of the event A in process P_1 is $C_1(a)$ and the time timestamp of event B in process P_2 is $C_2(a)$. So logical clocks can be thought of as counters which takes events as inputs and outputs the timestamp for that event. The happened before related is denoted by the symbol "$\rightarrow$" [49].

Lamport ordering of events has three main rules to logically relate events in a distributed system:

- If A and B are events in the same process P_i then $A \rightarrow B$ if timestamp of A i.e., $C_i(A)$ is less than timestamp i.e., $C_i(B)$
- If A is an event where process P_1 sends a message M and B is an event where process P_2 receives the message M, then $A \rightarrow B$ because one cannot receive a message before it is sent.
- If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$ (Transitive property) [8]

This happens before relationship creates only partial ordering among events, this means that only some events are related to each other between different processes. Events within a process are still related to each other within the same process. Each process uses a local counter which is an integer, a process increments the counter when a send event or an instruction execution event happens at it. The incremented counter or the logical clock value is assigned to that particular event as its timestamp. A send event carries its timestamp along with the message and then the receiving process updates its counter by 1 based on the max value of (received timestamp or the local clock value) [43].

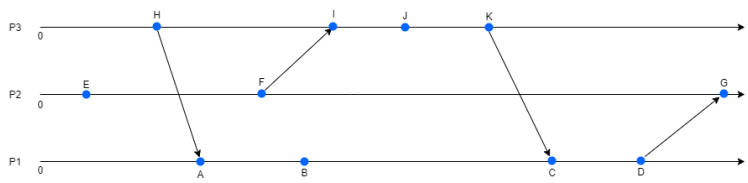

Figure 2.8: Processes and Events without Logical Timestamps

An example of how logical clocks work is shown using fig. 2.8, which consists of three processes with different events, and executions that are happening in and among those processes. An example of an instruction execution happening in process $P2$ is the event E, H is an example of sending event, while A is an example of receiving event. Initially all the processes start with counters with timestamp 0.

Following are the steps to assign timestamps to all the events.

- Process $P2$ executes an instruction E and assigns a timestamp of 1
- Process $P1$ sends the event H assigning it timestamp 1
- Process $P3$ receives the event A and checks the max (local timestamp, message timestamp) i.e., $max(0, 1)$ and adds 1 to it assigning timestamp 2
- Process $P2$ assigns F the timestamp equals to 2 by incrementing its local clock value by 1 and sends it.
- Process $P3$ receives the event I and checks the max (local timestamp, message timestamp) i.e., $max(1, 2)$ and adds 1 to it assigning timestamp 3
- Process $P1$ executes an instruction and assigns B a timestamp of 3 by incrementing its local clock by 1
- Process $P3$ executes an instruction and assigns J a timestamp of 4 by incrementing its local clock by 1
- Process $P3$ sends the event K assigning it timestamp 5 by incrementing its local clock by 1 and sends it
- Process $P1$ receives the event C and checks the max (local timestamp, message timestamp) i.e., $max(3, 5)$ and adds 1 to it assigning timestamp 6
- Process $P1$ sends the event D assigning it timestamp 7 by incrementing its local clock by 1 and sends it
- Finally, Process $P2$ receives the event G and checks the max (local timestamp, message timestamp) i.e., $max(2, 7)$ and adds 1 to it assigning timestamp 8

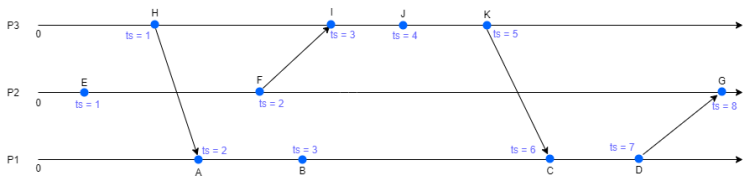

Figure 2.9: Processes and Events with Logical Timestamps

Fig. 2.9 shows how Lamports algorithm rules work to increment timestamps and assign them to events across the nodes in a distributed network.

Thus, having these logical clocks help in partial ordering of events to show causality across the network [49].

Chapter 3

Related Work

This chapter highlights some of the key researches that are performed in the area of wireless sensor networks, cryptography, effective memory management, and energy efficiency etc. To present the existing studies in an organized manner, the papers are categorized into various segments, which are provided in the following sections.

3.1 Research on Wireless Sensor Network

Multiple Base Stations

Deployment of multiple base stations is considered in order to reduce the overall energycostsinwireless sensor networks.This paper uses MRMS (Multipath Routing in Large Sensor Networks with Multiple Sink Nodes) , a protocol used to improve dynamic cluster maintenance, path selection and path switching in order to improve energy efficiency.A significant number of recentresearchesfocuses onoptimizingrouting protocols with the clustering approach, but not many studies have taken multiple base stations into consideration.While few researchers have considered the use of multiple sink nodes, there are still problems with saving residual energy from sensor nodes, path switching, ormanagingexcessive energy consumption.The reconstruction of the cluster is triggered by the nodes near the sink, as the energy consumption is higher with the nodes closest to the sink than the nodes far from the sink.Performance evaluation is performedon the basis oftime to first node failure, total number of dead nodes, mean energy consumption of one packet, average hop count to sink and packet delivery ratio.Based on the results, the proposed MRMS protocol greatly outperforms other protocols, nearly doubling or tripling the time to failure of the first sensor node.The study shows that MRMS substantially reduces the number of dead nodes and isactually moreenergy efficient.Particularly in comparison to the Voronoi

algorithm,TopDiskalgorithm and Direct Flooding, MRMS significantly reduces energy consumption.Exploring the impact of a loss MAC layer on the MRMS as well as building node-disjoint multipaths for multiple sink nodes is still required[17].

Base station mobility in wireless sensor networks has been introduced to maximize energy efficiency and lifetime of the network. This article provides a new method for data recovery in order to improve energy consumption and to boost network life through the use of Multiple Enhanced Specified-deployed Subsinks (MESS) process. Previous work [56] includes a routing protocol that achieves an extension of the network life, however each node needs to maintain the current location of the sink in real time that causes communication overhead to the sink as it needs to inform each node of the change in the network. Many studies have proposed different algorithms, some of which are not feasible at the moment or only applicable to particular WSNs. In this article, the suggested MESS system uses the installation of subsinks to collect data from other remote nodes. This can eliminate the node-to-mobile sink long-distance transmission and reduce the delay generated by long-distance. Results of simulation demonstrate that the strategy introduced outperforms conventional solutions and manages to save moreresources [79].

Multiple Base Stations and Increase Network Lifetime

Maximization of network lifetime is one of the most important design targets of WSNs. Data relaying to a static base station may extend the entire lifespan of the WSN, but the hotspot problem still exists. Having several base stations can help mitigate this issue with the hotspot problem and increase the lifespan of WSN even further. The researchers explore various mobility patterns in order to understand the optimal features for maximization of WSN lifetime. Different base station deployments are also discussed in this paper. In consideration of the same set of sensor nodes and base stations, three mobility patterns are compared. Regardless of the static or dynamic form of sinks, the lifespan of the network is significantly increased in all three patterns compared to the use of single base station. According to the tests, the performance of the network life is greater when spiral mobility is used in a single base station network where grid mobility is best suited to multiple base station deployments. The findings also show that there is limited difference in performance between multiple

static and multiple dynamic base stations. Hence, this paper presents various results and recommendations for better network performance [13].

Multiple Base Stations Placements

Alsalih et al. [6] aim to improve energy efficiency in a wireless sensor network by employing multiple mobile base stations. Numerous researches have been done to improve the existing routing protocols and satisfy the energy efficiency requirement; however, topology related problems are still in place. The advantage of multiple base stations is more reliable network load distribution. This paper primarily focuses on designing a base station placement scheme to distribute in the network, thus help to prolong the lifetime of the entire network, overcome bottleneck problems for the sensor nodes and faster delivery of collected data to the base stations. Base station placement is calculated based on the routing patterns and maximally overlapping regions. In this way, unlike other strategies and approaches, base station location varies over time by optimizing the network and prolonging the network lifetime. The results show that the developed approach even has the ability to double the network's lifespan.

It is desirable for sink nodes to be closer to sensor nodes, not just because sensor nodes lose significant amounts of energy to transmit other node data, but also fordisaster management. An effective multiple sink location strategy will significantly reduce issues such as shortening multi-hop distances between sensor nodes and sinks. Some previous works used grids to partition the network and used one sink node per partition, few others proposed using linear integer programming, iterative clustering methods, and random placements of sink nodes that are not very suitable for time-critical purposes. This proposed work considers number of sensors present, sink nodes present and forms grids. The sink placement is also based on geographic based method, candidate location with minimum hops and centroid of the nodes in a partition. The simulation results state that centroid of nodes in the partition (CNP) method performs very well compared to other algorithms and it even outperforms the GSP benchmark algorithm [67] which is used to solve the sink placement algorithm [22].

Multiple Base Stations For Key Management

Most of the current literature works consider a single base station WSN. Nevertheless, as the sensor network size increases, the distances between the base station and the corresponding sensor nodes increase which causes various problems. Most of the previous works use traditional key management algorithms to manage sensor nodes, but in the case of multiple base station systems, the preloading of a shared key with sensor nodes or the distribution of a pair wire key with a base station has several issues, such as the compromise of the base station making keys exposed. The majority of research studies assume that the base station is reliable and considers only the compromise of the sensor nodes, Nonetheless, key distribution and key revocation schemes are used in this design to prevent malicious network-connected nodes. The use of the node revocation list helps in situations of base station compromise, sensor node compromise or both. The proposed method does not add any additional communication overhead and the revocation time of the key is very well optimized [91].

Security of WSN with Multiple Base Stations

Due to the absence of good wireless network infrastructure, networks are exposed to security attacks and threats [4]. This paper by Alattas focuses on the black hole problem in wireless sensor networks. In this network attack, the intruder re-programs nodes to prevent them from relaying or sending any network traffic data further to the destination. In order to resolve this problem efficiently, multiple base stations and check agents have been used. The researcher adopted two different algorithms for providing efficient routing based on security risks, and this design also helps in saving energy. Check agents randomly visit nodes existing in the network and checks the frequency of packets obtained from neighboring nodes. This process helps in verifying if a neighboring node is acting as a black hole node. The use of check agents helped in identifying black holes attacks by 99%. The use of multiple base stations is proved to increase efficiency, security and also in the reduction of energy consumption.

Energy Efficiency of WSN

Many researches have been done in wireless sensor networks to reduce energy consumption of the sensor nodes using different new protocols and algorithms. The central focus of this work is to improve LEACH protocol and deploy multiple base stations to obtain better consumption of energy [71]. Several assumptions are made with sensor nodes and base stations and various properties like reachability, power efficiency and clustering interface are evaluated. Through increasing the number of nodes and the number of base stations, energy consumption per node is examined. Based on the results, a new design is proposed to enhance the protocol by replacing cluster heads with base stations, removing the leach protocol's limitations with only one hop communication.

Mobile Base Stations

In this scheme, the lifetime of the sensor network is divided into equal time periods called rounds, the base stations are moved for each round in order to conserve energy at each node of the sensor. The proposed protocol also helps to create new base station locations and aflow-based routing protocol. An integer linear system is used to assess the new locations for the base stations to be placed in the network, taking into account the energy spent on each node. It helps to reduce the total energy consumed in a round by the sensor nodes. The flow information obtained by solving the linear integer program can be used by sensor nodes to route messages in an energy efficient way. In order to calculate efficiency of the proposed system, different performance metrics such as time until anode dies, time untila percentage of nodes die, the amount of messages received and energy expended per round are considered. This paper suggests the energy-efficient usage design, for the reliability of wireless sensor networks withmultiple mobile base station systems [31].

This paper presents a new energy-efficient network model that uses Harmony Search algorithm to dynamically relocate a mobile BS to a cluster-based network infrastructure. Optimal CH's are selected from the sensor group to distribute the sensor roles equally. Efficient data transmission is extremely important given the large proportion of energy consumption due to relaying of the sensed data to the sink node or base station. Harmony search algorithm works on considering memory

factors, random consideration and pitch adjustment to create a new solution vector. This solution vector is improved by iterating the algorithm multiple times until an efficient optimal HMV is selected. This algorithm is responsible for creating a suitable network infrastructure that carefully selects the optimum number of clusters and their members. For every round, repositioning of the base station to the midpoint of all the CHs is considered which helps in minimizing the energy consumption. The results show that the proposed model obtained best score and outperforms other existing algorithms in terms of first node failure, last node failure and total packets sent to BS [5].

3.2 Research on Memory Allocation

Pathak [66] addressed the problem of developing efficient memory management strategies to enable new technologies that need more storage to handle real-time traffic. The researcher acknowledges that this memory management issue in wireless sensor networks has a significant gap in research and also identifies several constraints in developing effective memory management implementations. Memory management refers to various techniques for allocating and releasing memory blocks to different processes and threads on a system. Several problems are listed, including virtual memory, secondary storage management, small footprint, dynamic memory allocation, reprogramming and memory management. As more memory capacityis needed for new applications with real-time traffic, secondary storage for sensor nodes is considered because large amounts of data are captured and processed at the nodes, and this data have to be stored and retained. There is very limited research in memory management for multiple concurrent programs and support for virtual memory management in current WSN operating systems. Although substantial amount of work is completed, a broad range of research gapstillexist in this area [66].

Machaya et al. [57] aimed at working on reliable memory and storage management in wireless sensor nodes during the key generation phase. Current wireless sensor networks are a new formof distributed embedded systems with a wide range of applications in real time that need powerful memory management techniques. The authors addresseshow memory is handled by various operating systemsof wireless sensor networkslikeMANTIS,LiteOS,TinyOS, Nano-RK, LIMOS, SOS, Contiki, Enix

and μC/OS-II. However, the other studies do not discuss the memory management during key generation which is the primary focus of this paper. This research goal is accomplished by means of minimaxsampling, resulting in increased signal compression and improved memory management. The proposed solution reduces the number of samples and reduces the use of memory in the WSN node, however practical implementation of this approach on a real wireless sensor network is required.

The primary function of wireless sensor-networks storage protocols is to accurately duplicate and distribute data through nodes and to improve data collection and query via sink nodes. In network protocols are responsible to do this replication of data in the network. This article explains and illustrates the key principles of such current protocols that are further subdivided into multiple classes based ontopology, load balancing, transmission strategy and reliability. The main purpose of WSN is to collect environmental data and send it to users via a sink node, a powerful node connecting WSN to an external network like Internet. Many in-network storage systems were primarily designed for mobile sink management and node fault tolerance aiming to make information available to the users even in the case of node unreachability due to node damage or battery exhaustion. The assumptions by the researcher include, large scale wireless sensor network, limited resources, limited power, dynamic network and the use of in-network storage. Benefits and disadvantages of various approaches are discussed in terms of in-network storage requirements and additional challenges. All of these approaches rely on data replication utilizing sensor nodes, although not many research studies concentrate on the availability of data and network efficiency at base station level [60].

3.3 Literature on Security, Blockchain, Logical Clocks, and Data Storage

Elliptic Curve

Several experiments are carried out to determine the energy consumption of public key encryption in WSN. One of such research is by [83], where comparison of two asymmetric key cryptography algorithms, RSA and Elliptic curve cryptography is presented. Comparison is made regarding mutual authentication and key exchange

between two untrusted nodes within a network wireless sensor. The impact of public key encryptions on battery capacity and other variables that influence energy consumption are considered. Strong encryption will help secure communication and maintain confidentiality, completeness, authentication and non-repudiation. All these objectives can be achieved by incorporating symmetric key algorithms, asymmetrical key algorithms and hash functions. RSA algorithms are extensively used in many applications today in view of public key cryptography. Nonetheless, elliptic cryptography provides the same advantages with far less number of keys, which particularly helps WSN nodes to save more energy and to increase overall lifetime. The energy consumption to digitally sign a message using RSA-1024 is 304 whereas ECDSA-160 is 22.82 which shows that RSA consumes more than 13 times of energy compared to ECDSA providing similar level of security. This cost is more than 37 times to sign a message using RSA-2048 compared to ECDSA-224 having similar amount of security. This clearly shows that RSA costs way more due to the use of larger keys when signing messages. However, verification consumes more energy using ECDSA than RSA algorithm. In addition to this, key exchange operation is very efficient in ECDSA at server side and is almost the same at client side with negligible difference of energy. Hence, use of elliptic curve cryptography provides great computational and communication efficiencies [83].

ECC for WSN

WSNs can be run in large numbers and are often deployed in unattended environments where tampering, eavesdropping, altering information transmitted and adding unauthorised messages can be introduced in the network. Taking these attacks into account and to combat these threats, robust and effective security approach is necessary. Elliptic Curve Cryptography (ECC) enables this level of security to be achieved, as conventional security mechanisms are inadequate to provide much security in wireless sensor networks. This paper introduces a key management strategy to securely distribute secret keys to authenticated motes using ECC-based cryptographic algorithms. The proposed approach uses TinyECC implementation of elliptic curve encryption for wireless sensor networks that incorporates Elliptic Curve Digital Signature Algorithm (ECDSA), Elliptic Curve Diffie-Hellman (ECDH) and Elliptic Curve

Integrated Encryption Scheme (ECIES). This use of TinyOS implementation helps achieve scalability. Even if the network size increases significantly, the system requires little or no alteration. The time required to send a message using this approach shows 26% overhead, although the random number generator offers several advantages for resisting hardware attacks and provides great security against replay attacks. Thus, a robust ECC-algorithm security system has been designed for a reliable and efficient key distribution scheme [55].

Logical Clocks

The problem of timestamping and ordering of events in a distributed system can be solved using logical clocks. Leslie Lamport proposed this algorithm which is helpful in synchronizing a system with logical clocks and is useful in the ordering of events. Changing partial ordering of events to a consistent total ordering is the main objective of this system, which is achieved without using physical clocks, as they are not perfectly accurate. The concept of logical clocks uses "happened before" relation to create a partial ordering of the events in a distributed multiprocessing system. Five rules are defined by the algorithm for total ordering of the events with the help of partial ordering of events, which can be used to solve the problem of synchronization. As total ordering is considered somewhat arbitrary, additional use of physical clocks can prevent that problem. Hence, this paper benefits to understand the synchronization problem in a distributed system and a solution to solve the problem [49].

Merkle Hash for Commitments

This paper proposes a digital signature based on conventional encryption function which does not depend on difficulty of factoring and it does not require high computational costs. It clearly explains how one-way functions work, Lamport's one-time signature concept and its limitations are described. To overcome the limitation of signature size, Merkle proposed an improved version which is further improved by Winternitz's one-time signature. But the problem of signing more messages still exists as the storage and computational requirements are very high. This led to the proposal of an Infinite tree of one-time signatures now called as Merkle Trees. This algorithm requires very small amount of check signatures and is very fast compared

to other algorithms. Thus, digital signature system which works rapidly and requires less memory requirements is presented [61].

Blockchain Use

Different methods to protecting data are established using trust-wide encryption, such as cryptographic schemes and routing protocols. These approaches, however, are very critical in identifying the network's optimal path but unauthorized node attacks cannot be prevented. A special algorithm is implemented by combining the routing protocol AODV (Ad Hoc On-Demand Distance Vector) with the Particle Swarm Optimization (PSO) to produce trustable routing through the blockchain at all locations. Because the blockchain has an immutable digital economic transaction ledger that can record all financial transactions with their virtual values, if any breach happens through a node in the network, the other nodes will receive the alert details about this failure. The findings of the simulation work have shown how the AODV and PSO algorithms can be applied using the Poisson distribution probabilistic approach. This experiment has therefore contributed successfully to the distribution of information through blockchain, and the PSO algorithm helps to recognize congestion [7].

This paper proposes a blockchain trust model (BTM) in wireless networks for malicious node detection. Malicious nodes are one of the biggest threats in wireless sensor networks to launch attacks from within the network. Previous works may have the ability to detect malicious nodes effectively, however the detection process documentation is not stated, nor the mechanism to safely store the original data for later traceability. A new way to detect malicious nodes in wireless sensor networks is the advent of Blockchain technology and smart contract. This model's configuration has several sink nodes linked to multiple sensor nodes and a base station as the data destination. The architecture of the blockchain guarantees that the data cannot be changed, while the smart contract built on the distributed ledger is used to calculate the η value for the identification of malicious nodes. Many parameters are considered, such as delayed transmission, forward rate and response time. The results of the simulation show that the model can effectively identify malicious nodes in WSNs and ensure the detection process is traceable [75].

Several researchers continue to use different network development techniques and consensus algorithms to incorporate Blockchain technology in WSN and IoT. In this concept of rolling blockchain, smart cars were used as network nodes for building WSN. This paper suggests the process of block formation and structure in the chain and provides a statistical model for it. The existing projects recommend using external blockchains to pass transactions. As a consequence, customers are required to pay the fee for each transaction or pay the block formation chargeto miners. This paper focuses on building a private blockchain network and a node consensus protocol. It discusses the network structure, the chain structure, the block structure, the blockchain formation, the segmented network and the mathematical model of the proposed framework. Building a fixed sensor network and randomly spreading additional sensors until a certain density is attained to form a network. Shortest paths are then created using Monte Carlos algorithm by removing random paths. The test results revealed that even with an increase in attack density (increased number of lost connectionsand nodes) the network remains stable and the Blockchain can stillbe built. The security analysis and hacking problems of the proposed method have not been addressed. Not many studies are carried out to apply the findings of this article to the sensor networks and the Internet of Things [48].

Most research focuses on integrity, privacy, confidentiality, key management, efficient routing, secure data storage and availability of wireless sensor networks. Only few such research focuses on recovery of failure nodes. This paper uses WSNs node recovery scheme based on blockchain to first identify the failed nodes using cluster heads status. Only through the cluster head can sensor nodes or non-cluster heads communicate with the sink node. The cluster headscontinue to track the stateof the other connected nodes. Smart contracts are used to initiate the cluster head's recovery process, which in turn helps with node recovery. In the case of working CH, a consensus algorithm will be executed between the CHs to ensure that state is in place. If 51% of the consensus for CH is reached, it receives data from the sensor nodes and forwards it to the sink. If a failed cluster head is detected, recovery process starts by selecting the replacement for the failed CH. The security analysis of the proposed framework is carried out in order to ensure high security and cost analysis shows that the resulting cost is very low in order to recover the node in the network [80].

Using Blockchain for Data Storage

Reliable data storage and support is very important when it comes to wireless sensor networks. As the storage space increases and it requires energy from each node to hold data and provide its storage, efficient incentive mechanism supports and promotes nodes to store more data. This paper proposes an incentive mechanism which gives incentives in the form of real money based on the amount of data stored by the nodes. It also focuses on how to ensure normal operation of the network when there is a data conflict among the nodes and to eliminate those false nodes. Many such researches proposed different solutions to this problem of limited storage capacity. Data storage capacity is an important resource, which usually is performed using external storage, local storage or data centric storage. Blockchain based incentive mechanism is proposed to encourage nodes to store data. This system has different layers in order to achieve various services like data encapsulation, incentive mechanism, network consensus, and programmability. Instead of the proof of work algorithm to achieve consensus, Provable Data possession mechanism is used which needs very less computational power when compared to proof of work mining algorithm in the bitcoin protocol. This way, the proposed design takes the advantage of blockchain protocol eliminating its limitations by replacing pow with PDP mechanism and promoting data storage in a wireless sensor network [73].

Using Blockchain for Encryption and Security

The motivation behind this proposal is to design a complete model which works for security, privacy, trust management and node authentication in a sensor network. Previous works on these areas have developed several solutions but none of them satisfy all these requirements altogether. With the presence of authentication and trust management issues in wireless sensor networks due to its security constraints, a design with blockchain data structure helps in storing decentralized authentication and trust management information. This paper proposes one such model which is adaptable, and it ensures reliability over time. Blockchain Authentication and Trust Module (BATM) is designed which provides confidentiality with encryption, authentication using digital signatures, identity validation with the help of peers. Different payloads are present in this model which is used for different purposes such as miner

approval to validate the payload, credential payload to approve a node, renew payload to renew credentials, blame payload and revoke payload. As trust is a probability, it needs to be evaluated which is carried out with the use of reputation factor, this helps to decide whether to allow or restrict actions in the network by the network node. However, this model still needs to be improved by considering the results of every BATM part reputation. Thus, a new design for secured decentralized storage of cryptographic keys, trust information using blockchain in a wireless sensor network is proposed [64].

Even with the rapid growth in research and technology for IoT, the problem of traditional security and privacy still exists. Blockchain has increasingly been used to establish privacy and security in peer-to-peer platforms with identical systems to IoT. This blockchain technology is used as the providing solution in this paper on the field of IoT by taking smart home architecture as an example. Regardless of blockchains limitations when it comes to latency, resource intensiveness, and scalability, Dorri et al., introduces a lightweight and decentralized blockchain based architecture for IoT. Blockchain is mined and maintained by one or more resource capable devices like smart hubs or home computers. The homeowner controls and maintains any smart home device and an overlay network is used to connect to other networks externally. There may be an optional local storage for local data storage in each home or they may use cloud storage if they want to store information on cloud. Homeowner also has the option to use the miner's monitor transaction to track and access information. Not only does the blockchain store data, it also stores access information and policies to allow or reject network transactions. Unlike the blockchain used in Bitcoin, this conceptual design does not use any consensus mechanism as it is private blockchain and is operated by the homeowner. Performance of the proposed architecture under specific security and privacy risks was assessed and qualitatively analyzed [26].

Synchronization Problem

When using existing client server model with thousands of IoT devices connected to each other, synchronization issues are expected, and the server client model may have some limitations and issues. Therefore, the research uses blockchain architecture to monitor and customize IoT devices instead of the client-server model. Ethereum,

which provides Turing complete programming, allows developers to write smart contracts and has a high speed. Ethereum is used by the proposed design to control and configure a group of IoT devices. This model uses a smart phone and three Raspberry Pis where each one of them is viewed as an air conditioner, Light bulb and a meter for monitoring electricity use. There are three smart contracts built on the Ethereum blockchain for tracking electricity usage, writing policies and key management functions. The results indicate that, even with Ethereums high speed, it is still not fast enough and the absence of a lightweight client makes it prone to security problems. Thus, this experiment proposes a new way of managing IoT devices using the Ethereum blockchain platform [37].

Use of Blockchain and other Peer-to-Peer Technologies to Increase Distributed design

The research objective in this paper is to understand if the approaches to blockchain and peer-to-peer can be used to advance distributed and private-by-design IoT. Various blockchain implementations have been found in the literature in which four of them are IoT related. This entire study is based on Kitchenham's SLR guidelines [45]. Requirements such as blockchain uses outside cryptocurrency, IoT applications, implementation discrepancies with bitcoin blockchain, degree of integrity, privacy and flexibility are taken into account. From eighteen blockchain uses beyond cryptocurrencies, four of them are linked to IoT. Data storage location and distinct techniques used for blockchain mining are identified. With regards to blockchain integrity, different attacks found in the papers have been analyzed to ascertain the vulnerability of blockchain. The security risks include significant problems in reaching consensus, deliberate forking of the blockchain, delay in publishing mined blocks to erase and replace existing chain, creating a new branch via historical revision attack and so on. Four classes of de-anonymization strategies various adaptability issues are recognized. There are thus many applications for blockchain, but it is viewed as less ideal for IoT with the scalability issues [20].

Comparison of Blockchain with Centralized Network

Kumar et al. [47] discuss potential security and privacy issues related to information exchange and data authentication via central server, and examine how this is solved withthe blockchain-based distributed ledger network. As the number of devices increases, communicating using a centralized network can result in many problems. In order to understand the role of blockchain technology, different issues related to IoT and IoT with BC were also addressed. Interaction between the three elements i.e., Networked Sensors and Actuators (NSA), Raw Information and Processed Data Storage (R-IP-DS), Analytical and Computing Engines (ACE) has been analyzed to examine the risks of security and privacy issues arising. A comparison of data flow is made using a centralized server and Blockchain technology. To large-scale IoT applications, blockchain technology has the following benefits such as tamper proof data, trustless and peer to peer connection, robustness, high reliability, accelerated transactions. Category-based blockchain implementations in various sectors and different usage cases are clearly shown. This paper therefore provides a basic idea of the need for blockchain in IoT.

Blockchain Smart Home Framework to Increase Confidentiality, Integrity, and Availability

Safety and privacy of the Internet of Things (IoT) is a significant challenge, especially due to the vast size and the deployment of IoT networks. With the purpose of ensuring confidentiality, integrity and availability, the proposed blockchain Smart Home Framework is thoroughly evaluated through its security objectives with simulation results which illustrate the overhead costs. Through eliminating the use of the mining algorithm POW (Proof of Work), this architecture relies on hierarchical structure and mutual trust to preserve the security of blockchain and is compatible with IoT. The key smart home components are transactions which are used to interact between local nodes and external networks, local blockchain processed and stored by local storage, and then a home miner managing this blockchain. In this design, the dedicated home miner plays a significant role from initialization to transaction handling and shared overlay communication management. This design is evaluated on the basis of security and performance aspects. In terms of security, malware deployment in smart devices

is unlikely because they are not directly accessible, eliminating the possibility of various attacks, symmetric encryption is used for confidentiality, hashing is employed to ensure integrity and policies are needed for authorization. For performance measurement, packet overhead, time overhead as well as energy consumption are used. The results show that this blockchain design for smart homes involve low overheads and that the protection and privacy benefits are worth their weight [25].

Using Blockchain Technology for Immutability of Data

In order to make sensor data unmodifiable and permanent, thus increasing availability of the data, Ibba et al. [38] have adopted blockchain technology in their urban smart city sensor network. Due to blockchain's property of immutability, security and transparency, it is very much suitable in this smart city environment monitoring system. Mobile devices and sensors are used to connect and modify the blockchain by adding information, providing reports, feedback from users. Data is collected by various sensors and is sent to the collection end point by distributed sensors, this data is validated by a specific mining software. Sensors send data themselves or by using mobile devices to the blockchain. Ethereum system is used as the blockchain technology in this paper. Thus, from this blockchain network, users can access information about the environment.

Black Hole Problem

The crucial issue for wireless sensor networks is how reliable data collection can be done under different conditions. Many captured sensor nodes in wireless sensor networks (WSNs) are converted into black hole nodes by tapping, destroying and shielding information received. Most of the sensor nodes near the sink node are captured to avoid the information being sent to the destination. First, the construction of secure multi-path routing is done and then the secure collection of data is done on a secure multi-path basis. Based on the simulation performance, the authors claimed that the proposed algorithm does not ignore the node compromise near the sink node, unlike other algorithms. The design of MERS (Multi-Error Report Simultaneously) helps the sink node to reuse the number of data items which are incorrectly received. Thus, a feedback-based algorithm to implement a secure multi-path data collection routing

has been presented [84].

Use of Storage Nodes

A large amount of data has been collected for future data query and analysis in most wireless sensor network applications, so how to store them becomes a major challenge in such wireless sensor networks. A kind of storage node has recently been adopted as a useful strategy for solving the storage challenge, but storage node placement has become a critical issue in such wireless sensor networks. The aim of this algorithm is to balance wireless sensor energy in each storage node and to reduce a performance measure called total energy costs, and to provide the most efficient solution to the energy consumption of the wireless sensor networks. Data is typically stored by either the sink node or the sensor nodes. Data request processing is much more difficult when the data is stored in sensors compared to the data storage in the sink node. But if all the data needs to be present in the sink node, it needs to travel very far from the sensor nodes all the way to the sink node and this process may use intermediate nodes to relay the data which takes even more energy. The positioning of the storage node problem is resolved in a fixed tree model rooted at the sink. This paper uses sensor nodes, storage nodes, as well as forwarding nodes for the tree T. The storage node is configured in such a way that the cost of sending processed data plus processing costs is lower than sending large raw data. The results showed that this algorithm helps to reduce the overall cost of energy by putting a storage node and has additional advantages when using the storage node [86].

Data Storage Using Routing Algorithm

The sensors need to use multi-dimensional range queries to retrieve data with range values and various attributes. This paper proposes a new storage method for multi-dimensional attribute data. Using a geographic hashing algorithm, multi-dimensional space is mapped to a 2-d geographic range space that helps address multi-dimensional range queries. First, the design of the distribution index for multidimensional data is completed. It involves the insertion of data that stores observed events at appropriate nodes in a corresponding range space, the query processor then analyses the user query and can subdivide the query into multiple sub-queries. A hashing algorithm is used to

map an event to an appropriate range of space in the data insertion phase. To store events in appropriate nodes this approach makes use of a routing algorithm. In the second phase of the query processing mechanism, in order to get the matching events, query resolution and recovery of events are performed. The tests of the simulation are performed using average cost of insertion and average cost of querying. The proposed data storage and range query system reduces the cost of message and hotspot problems even with the presence of a large number of events and sensor nodes in the network to achieve load balancing [51].

Chapter 4

SEA (Security, Efficiency, Availability) Framework

In this chapter we propose a new approach for the replication of data between base stations in a large dynamic wireless sensor network. This approach is motivated by a distributed ledger technology called Radix. We begin with an overall view of the topology and the assumptions made for the network, base stations and sensor nodes. Using the proposed scheme, we demonstrate data transmission and storage mechanisms in base station. The proposed framework efficiently works for both static and dynamic wireless sensor networks. Besides these, various schemes are proposed to achieve integrity and freshness of data are addressed.

4.1 Network Assumptions

4.1.1 Sensor Nodes

A sensor node is an intelligent, small sensor that can collect information, process and interact with other connected network nodes. The proposed system includes a number of sensor nodes evenly distributed for sensing data in a large network area. Sensor nodes are homogeneous and have very limited memory to store sensed data. Because the sensor nodes are computationally constrained and have a small range, data is transmitted by relay nodes or multi-hop communication to the nearest base station. Information is periodically sensed and also when the basestation requests current data.

4.1.2 Relay Nodes

These nodes are primarily responsible for transmitting / relaying sensor data from other sensor nodes to the closest base station. Data from various sensor nodes are collected to send it to the nearest connected base station (Fig. 4.1). Similar to sensor nodes, relay nodes are powered by battery and have the capability of communicating

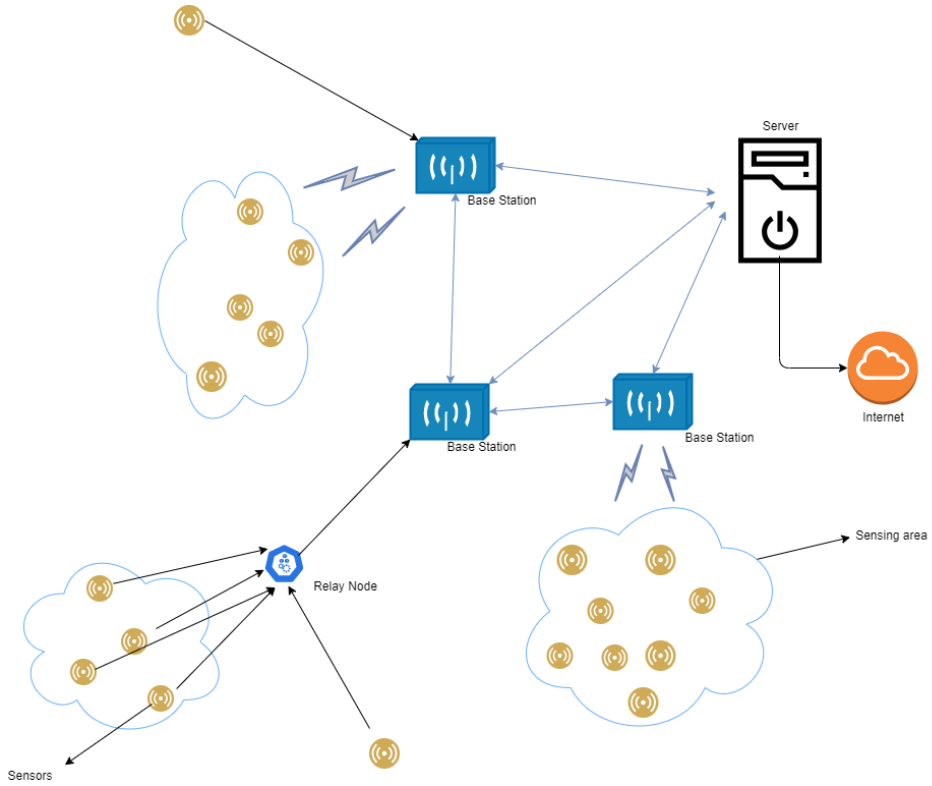

Figure 4.1: Wireless Sensor Network setup with sensor nodes, relay node(s), and multiple base stations

wirelessly.

4.1.3 Base Stations

Sensor nodes need a collection point where the sensed data can be analyzed, stored, or distributed to other networks through wireless communication systems of longer range and greater-throughput. Base stations often referred to as sink nodes, are mainly responsible to collect this sensed data in the wireless sensor network. The proposed architecture is designed for sensor networks with mobile base stations and also static base stations. It functions as gateway between the sensor node and the user, when requested, by sending data. The network has multiple base stations and has a higher memory capacity, computational power and battery life. Each base station communicates and replicates data with a set ofother base stations. We assume that there are N numbers of base stations on the network. We also believe that the base stations in thisprotocol are mobile and can switch from one place to another. Mobile

base stations are considered to have more power but are considered to be battery operated. They are also considered to have very long range, all the base stations have the capability to communicate with the server irrespective of its location. However, if a network having base stations with limited range is considered, base stations can request its neighbouring base stations to send the related data in its remote memory to the server, thus achieving data availability in such situations.

4.2 Research Scope

The scope of the proposed research does not include configuration setup between sensor nodes and base stations. There are, however, several other methodologies and techniques for efficient and secure communication between sensor nodes and base stations. The sensor data storage allocation in the base stations local memory take place when a sensor node is connected and starts storing the data sent by the sensor node. If it is disconnected from the base station due to number of factors that could be related to sensor node failure or sensor node connecting to another base station, the related existing sensor node data will be offloaded to the server and this change is reflected in the connected neighbour base stations replica memory.

From a technology point of view, Blockchain is used specifically to eliminate the concept of centralization and to achieve a distributed agreement. This requires security across the network, but the key concept is to get the same data across the network that is accomplished by replicating the ledger across all nodes as several copies. Such identical multiple copies are important for achieving consensus, security of that allows the information to be consistent. Actual data without manipulation is perceived to be more essential in the area of distributed ledger technology. The proposed protocol considers this replication and other security requirements as a crucial aspect, however the analysis of confidentiality and data integrity in regards to blockchain is not tested.

4.3 Secure Topology Setup

4.3.1 Configuration Phase

We consider n number of base stations $(\beta_1, \beta_2, ..., \beta_n)$ randomly deployed in the network. Every base station (β_i) will have its own public, and private keys $(pub_{\beta_i}, priv_{\beta_i})$

generated using a standard assymetric key generation algorithm, such as elliptic curve cryptography. Upon generating a pair of asymmetric keys, a base station (β_i) broadcasts its public key to all the other nodes. In addition, each base station generates an initial symmetric key (sk_{β_i}), which is updated at later stage. The symmetric key is used to setup connection between neighbouring base stations. Using the asymmetric keys, a base station (β_i) tries to send a connection request (Eq. 4.1) that contains its symmetric key (sk_{β_i}) to other $(n-1)$ number of base stations. Once n initial symmetric keys are created in the network, every base station encrypts the key one by one using the public key of the destination base station and sends it to that base station to establish a connection. The proposed approach uses elliptic curve cryptography to securely communicate symmetric keys between base stations. These established symmetric keys are then used for sending and receiving data between connected neighbouring base stations. The process of key establishment between base stations is done in this configuration phase. The number of base stations connected to each other depends on the replication factor which is also responsible to manage different number of data replicas in the sensor network. This number is predefined into the base station when it is deployed.

$$\begin{aligned}
reqConfig(\beta_1, \beta_2) &= ENC_{pub_{\beta_2}}(sk_{\beta_1}||ENC_{priv_{\beta_1}}(sk_{\beta_1})||h(location)) \\
reqConfig(\beta_1, \beta_3) &= ENC_{pub_{\beta_3}}(sk_{\beta_1}||ENC_{priv_{\beta_1}}(sk_{\beta_1})||h(location)) \\
reqConfig(\beta_1, \beta_4) &= ENC_{pub_{\beta_4}}(sk_{\beta_1}||ENC_{priv_{\beta_1}}(sk_{\beta_1})||h(location)) \\
&\vdots \\
reqConfig(\beta_1, \beta_n) &= ENC_{pub_{\beta_n}}(sk_{\beta_1}||ENC_{priv_{\beta_1}}(sk_{\beta_1})||h(location))
\end{aligned} \tag{4.1}$$

As indicated in eq. 4.1, the connection request message sent from each base station to the other base stations contain initial symmetric secret key (sk_{β_i}) encrypted with private key of source base station. This acts as a digital signature of the source base station (β_i). The hash of location $h(location)$ of the source is also sent as part of the connection request message. The time at which a base station (sk_{β_i}) sends this broadcast to other base stations is denoted as δt_s and the time of the response message received from the destination base station is denoted as δt_d. Each source base station computes the time difference (Eq. 4.2) to identify the nearest base stations

or available base stations. This is used in order to choose neighbours for connection establishment. The fig. 4.2 illustrates the connection request message to other base stations.

$$\delta t_{diff} = \delta t_d - \delta t_s \tag{4.2}$$

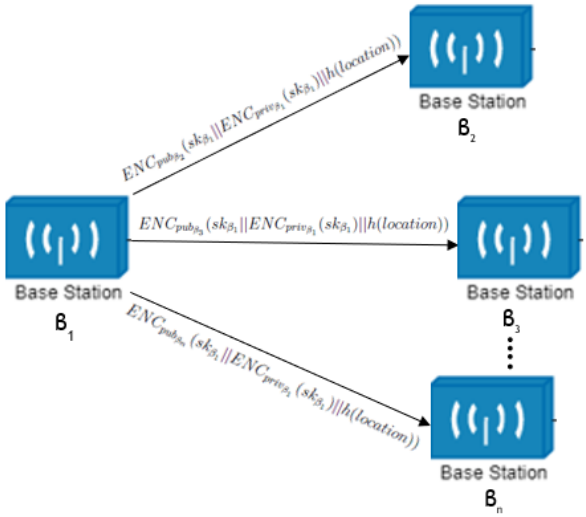

Figure 4.2: Sending request message from every base station to all other base stations

In general, SEA framework is not suitable for a highly dynamic network due to configuration overhead. However, high network speed, resourceful base stations and sensor nodes can minimize the configuration setup time.

4.3.2 Neighbour Selection Phase

The request (Eq. 4.1) from source base station is decrypted by all the destination base stations using their private keys. Each destination base station checks its remote memory to determine, if it is storing r number of other base stations local memories. The parameter r is the replication factor, which is configured at the time of the initial deployment of the network. If there is no room for adding another remote base station's data, then the destination base station will not send any reply to the request message. If remote memory is available, then it sends a response message (Eq. 4.3) to the source base station as an acknowledgement. The response message is generated by performing logical XOR of obtained initial symmetric secret key (sk_{β_i}), and the source

base station's hash location ($h(location)$). The XORed parameter is further encrypted with the private key of the base station to be used as the digital signature of the remote or destination base station. Since the network is dynamic in nature, the location value changes, and that minimizes the probability of replay attack. Fig. 4.3 shows the reply message acknowledging the confirmation and connection establishment by some of the destination base stations.

$$\begin{aligned} resConfig(\beta_2, \beta_1) &= ENC_{pub_{\beta_1}}(ENC_{priv_{\beta_2}}(sk_{\beta_1} \oplus h(location))) \\ resConfig(\beta_3, \beta_1) &= ENC_{pub_{\beta_1}}(ENC_{priv_{\beta_3}}(sk_{\beta_1} \oplus h(location))) \\ &\vdots \\ resConfig(\beta_m, \beta_1) &= ENC_{pub_{\beta_1}}(ENC_{priv_{\beta_m}}(sk_{\beta_1} \oplus h(location))) \end{aligned} \tag{4.3}$$

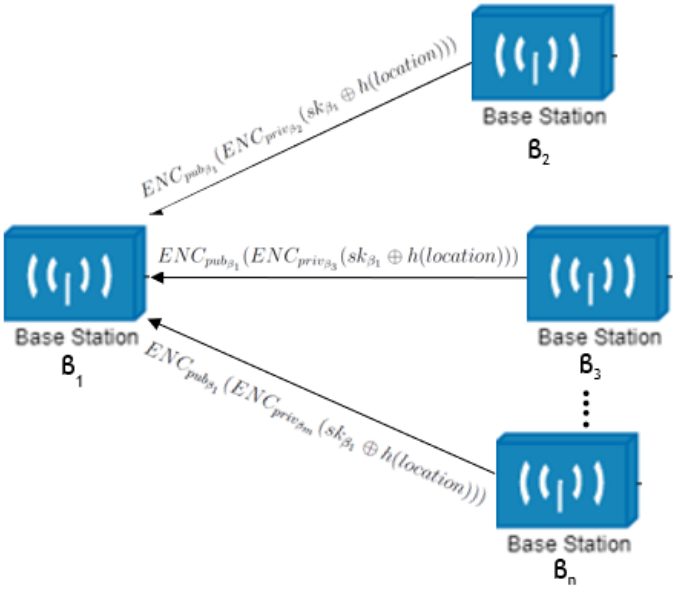

Figure 4.3: Response message from some remote base stations to the source base station

When the source base station gets acknowledgement from Destination base station, it decrypts the received message using its private key, $priv_{\beta_1}$. The source base station (β_i) confirms the signature of destination base station by decrypting the data using destinations public key to get the XORed message ($sk_{\beta_1} \oplus h(location)$). The source computes the message and compares with the obtained message. This process helps to determine the security and integrity of the message.

Now, the source base station computes the time difference between the configuration request sent and receive (Eq.4.2), and selects r number of neighbours based on r number of minimum δt_{diff} values. The source base station (β_i) sends its acknowledgement to the selected neighbours by encrypting the acknowledgement message (Eq. 4.4) with the public key of the neighbour base stations. δt_{diff} is sent as part of the acknowledgement, which is used later in the key update phase.

$$\begin{aligned} ackNeighbour(\beta_1, \beta_2) &= ENC_{pub_{\beta_2}}(ENC_{priv_{\beta_1}}(h(location))||\delta t_{diff}^{\beta_2}) \\ ackNeighbour(\beta_1, \beta_3) &= ENC_{pub_{\beta_3}}(ENC_{priv_{\beta_1}}(h(location))||\delta t_{diff}^{\beta_3}) \\ &\vdots \\ ackNeighbour(\beta_1, \beta_k) &= ENC_{pub_{\beta_k}}(ENC_{priv_{\beta_1}}(h(location))||\delta t_{diff}^{\beta_k}) \end{aligned} \tag{4.4}$$

4.3.3 Key Update Phase

Periodically, for every time t base stations need to update the established symmetric keys which are used to connect to other base stations. This time t is set during the configuration phase of the system. Every base station (β_i) initiates this key update request to increase security by updating keys periodically and ensure secrecy of the network connections. During this phase, every base station (β_i) (where $i = 1$ to $n-1$, n is the number of base stations) tries to send a key update request to connected base stations (β_k), where k is a value between 1 to $n-1$ and $k \neq i$. The key update request message (Eq. 4.5) contains digital signature of source base station, which is created by encrypting the XORed hashed location of source base station and the δt_{diff}. This message is encrypted with the public key of the destination base stations before sending it to the destination base stations (Eq. 4.5).

$$\begin{aligned} reqKeyUpdate(\beta_1, \beta_2) &= ENC_{pub_{\beta_2}}(ENC_{priv_{\beta_1}}(h(location) \oplus h(\delta t_{diff}^{\beta_2}))) \\ reqKeyUpdate(\beta_1, \beta_3) &= ENC_{pub_{\beta_3}}(ENC_{priv_{\beta_1}}(h(location) \oplus h(\delta t_{diff}^{\beta_3}))) \\ &\vdots \\ reqKeyUpdate(\beta_1, \beta_k) &= ENC_{pub_{\beta_k}}(ENC_{priv_{\beta_1}}(h(location) \oplus h(\delta t_{diff}^{\beta_k}))) \end{aligned} \tag{4.5}$$

Upon receiving the key update message, the neighbour or destination base station

generates a key using "Intermediate Key Generation Process". The source base station generates the same key after receiving the key update acknowledgement message (Eq. 4.6) from neighbour base stations. Both the neighbour base station and the source base station use "Intermediate Key Generation Process" to generate the same symmetric key.

$$\begin{aligned} ackKeyUpdate(\beta_2, \beta_1) &= ENC_{pub_{\beta_1}}(ENC_{priv_{\beta_2}}(h(location))) \\ ackKeyUpdate(\beta_3, \beta_1) &= ENC_{pub_{\beta_1}}(ENC_{priv_{\beta_3}}(h(location))) \\ &\vdots \\ ackKeyUpdate(\beta_k, \beta_1) &= ENC_{pub_{\beta_1}}(ENC_{priv_{\beta_k}}(h(location))) \end{aligned} \tag{4.6}$$

Intermediate Key Generation Process

In an intermediate key generation process, the source base station, and the destination or neighbour base station generates a symmetric key without sharing any parameters. This ensures minimal predictability of the key. If the intermediate key generation phase is invoked for the first time, then it replaces the first key (sk_{β_i}) generated by a source base station (β_i) with the newly generated key $sk_{\beta_i\beta_j}$, where β_i represents source base station, and β_j represents destination base station. After the initial key update, $sk_{\beta_i\beta_j}$ is replaced with the newly generated key $sk'_{\beta_i\beta_j}$.

To generate the identical symmetric keys at both ends the source base station (sk_{β_i}), and destination base station (sk_{β_j}) use PRNG (pseudo random number generator). During the first key generation sk_{β_i} is used as the seed value for the PRNG. From the output stream, a particular sequence of bits identified by $h(\delta t_{diff})$ is considered as the $sk_{\beta_i\beta_j}$ key (Eq. 4.7). For the subsequent key generation $sk_{\beta_i\beta_j}$ is considered as the seed (Eq. 4.8). Changing the seed ensures variability in the generated keys.

$$sk_{\beta_i} \xrightarrow[seed]{PRNG} OUTPUT\_STREAM_{h(\delta t_{diff}^{\beta_j})} \tag{4.7}$$

$$sk_{\beta_i\beta_j} \xrightarrow[seed]{PRNG} OUTPUT\_STREAM_{h(\delta t_{diff}^{\beta_j})} \tag{4.8}$$

4.4 Data Transmission and Store

4.4.1 Sensing Phase

After the configuration phase is setup, sensors will have established a symmetric key with the nearest base station, this key is then used to transmit data to the base station either directly or through relay nodes. Sensor nodes sense data and send this data periodically or when queried to get the current data.

4.4.2 Memory Management

Memory management in the proposed protocol is focused on managing base stations memory. Base stations memory is split into two different virtual partitions. One for local memory and another for remote memory (Fig. 4.4). Local memory is the memory where data related to all connected sensor nodes is stored whereas remote memory is the replica memory used to store other base stations data based on the number of replicas it need to store which is the replication factor. The figure (Fig. 4.4) shows how the partition is done in base stations for storing connected sensor nodes memory and providing storage space for remote memory storage of other base stations local memory.

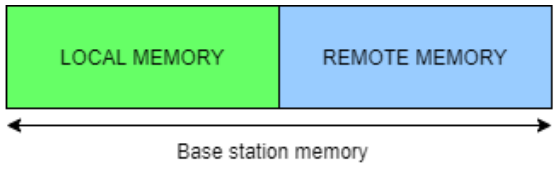

Figure 4.4: Base Station Memory Distribution

4.4.3 Replication Factor

We propose a dynamic replication strategy that meets both availability and performance specifications while taking the memory requirements into consideration. Distributed storage needs are growing as WSN produces large volumes of data from different sensor nodes. The duplication and distribution of data is a well-known method to achieve availability and efficiency in many other applications. This makes several copies of data in different locations of the network. In the case of failure of

nodes, at least one copy of the data will be available and present in the network, thus achieving fault tolerance and increasing the availability of data. The suggested solution does not overwhelm base stations with large volumes of redundant data but maintains service availability. This replication factor value is decided before the base station gets deployed. After the configuration is done, connected base stations share a part of data. When a base station connects to other base stations, the local memory of the source base station is stored in remote memory of the replication base station. As local memory only has connected sensor nodes data, the data replication is partitioned base on the sensor nodes. This replication of data is among the whole network which replicates the sensor data from one base station to multiple base stations. The approach suggested discusses the following issues of (i) what sensor data should be replicated, (iv) what is the replication factor considered, (ii) To which base stations should the data be replicated and (iii) when will data replication process start.

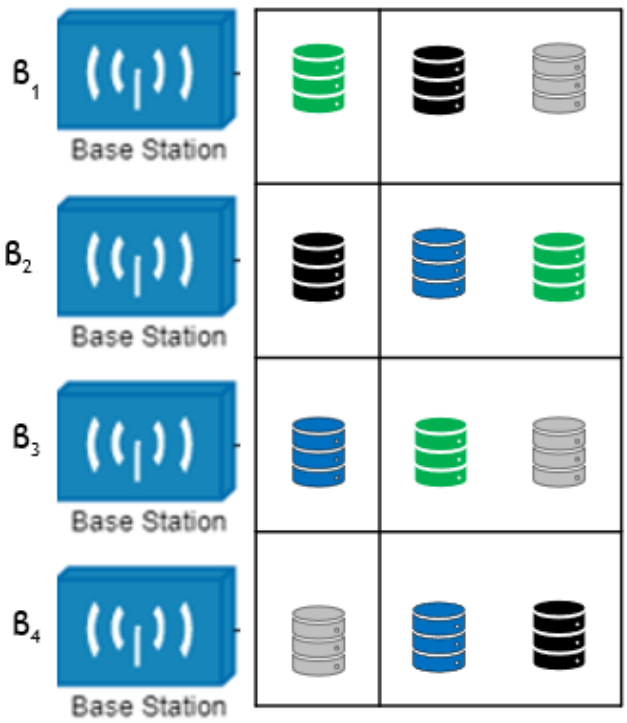

Figure 4.5: Base stations memory with replication factor 3. The memory on the left is local memory and on the right is remote memory. Local data is replicated in the remote memory of connected neighboring base

Data from sensor nodes are stored in the local memory of connected base stations, the goal is to replicate this data to other base stations. Replication factor (r) is the number considered before the network is deployed. It can vary from base station to base station. For simplicity, we considered the same replication factor for every base station. However, variation of replication factor provides varied rate of replication among the network. Base stations reply to those $(r-1)$ connection requests having minimum time difference (δt_{diff}). These base stations are used to send the data to create replicas of the source base stations local memory. The process of sending data to other base stations is dynamic. Base stations try to always keep their replicas updated by sending new data to achieve data freshness. Fig. 4.5 shows how data from one base station is replicated to other base station with replication factor 3.

4.4.4 Synchronization

Use of logical clocks helps in achieving data synchronization. For every sensor node connected to the base station, the base station keeps track of the sensor node information using a value called logical clock (Fig. 4.6).

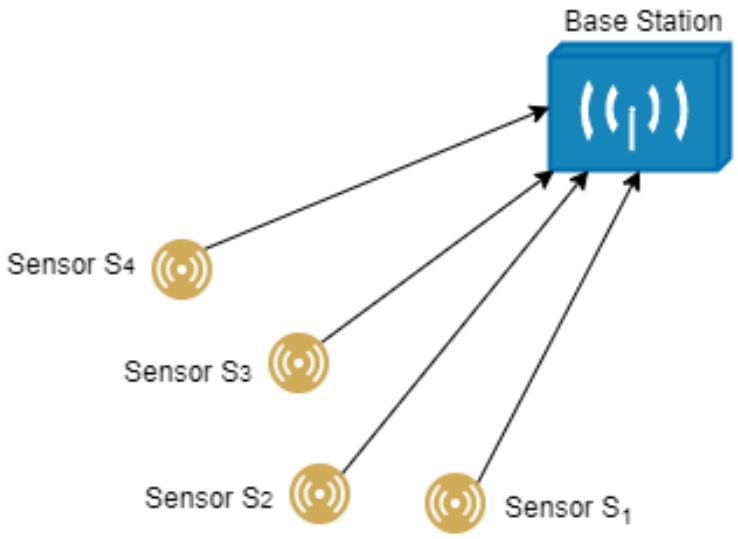

Figure 4.6: 4 sensor nodes communicating with 1 base station in WSN

When a sensor node senses data and sends it to the base station for the first time, this information is stored in the base station with logical clock value 0. This logical value is incremented every time a sensor node sends information about the current

sensed value and is stored with the new logical clock value appended to it. If base station sends this data as a replica to other base stations, it also sends the logical clock value to the other base station. The destination base station will keep track of the incoming sensed data replicas and checks the logical clock value order. This helps in determining, if any sensed data value is missing.

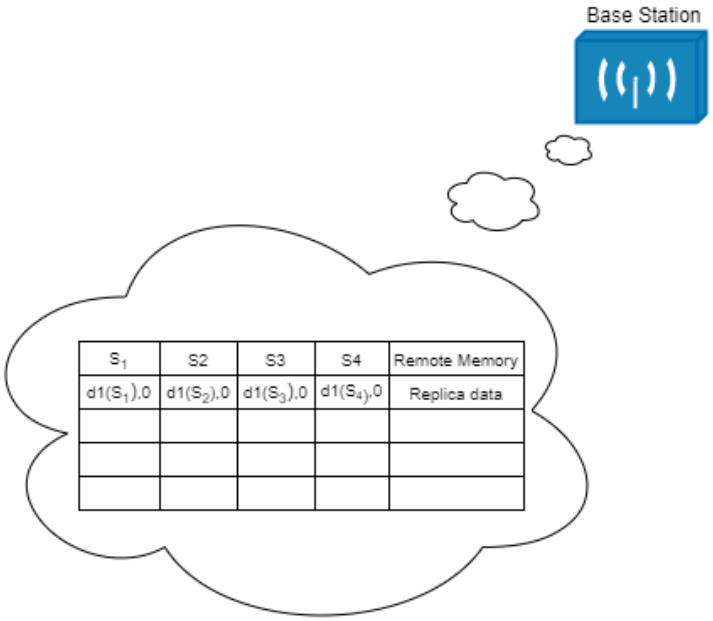

Figure 4.7: 4 sensor nodes connected to a base station β_1. For every sensor node s_i where $i = 1$ to 4, base station β_1 keeps independent logical clocks.

Considering 4 sensor nodes connected to a base station β_1. For every sensor node s_i where $i = 1$ to 4, base station β_1 keeps independent logical clocks. If every sensor node s_i sends information for the first time, base station updates its local memory as shown in fig 4.7.

Local memory of base station contains data about 4 connected sensor nodes from s_i where $i = 1$ to 4. The first sensed data by those sensor nodes is sent to the base station through relay nodes or directly. This is received by the base station and is appended with an independent logical clock value for every base station that starts with 0. The term $d1(s_1)$ denotes the data sent from the sensor node s_1 which is appended with the logical clock value 0 as it is the first data received for that sensor node. Further data received by the sensor nodes is added to the local memory of the base station with a new logical clock value incremented by 1. This helps in

uniquely identifying every value of the sensor node data and can be helpful in data synchronization with other base stations. Fig. 4.8 shows how the logical clock value is incremented by 1, and the new sensor data is stored in the local memory of the base station appended with the new logical clock value.

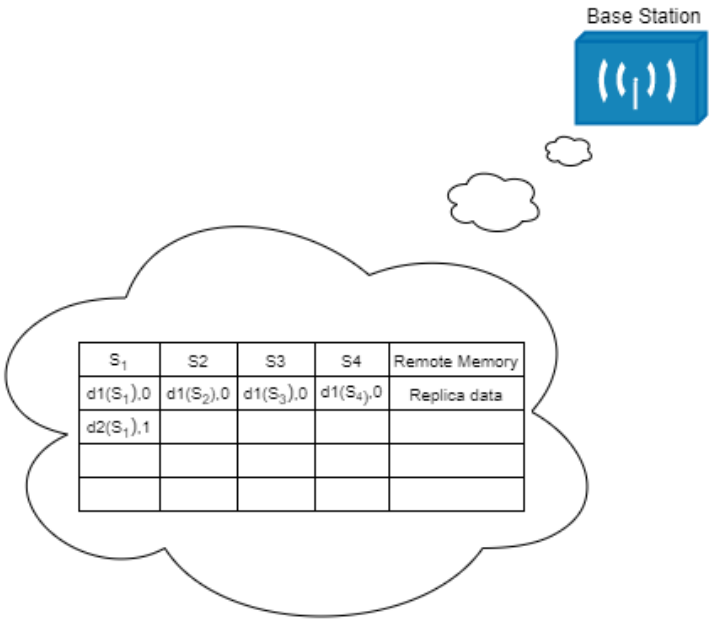

Figure 4.8: Increment of logical clock by 1. New sensor data is stored in the local memory of the base station

4.4.5 Commitments

Commitments is the concept of verifying replica data by the source base station. The proposed approach verifies its data by sending periodic commitment request to the destination base station where the data is being stored. This commitment request contains all the sensor ids present in the local memory of the source base station i.e., connected to the source base station and signature of the base station generated by encrypting the location of source base station using its private key. The fig. 4.9 shows the commitment request (Eq. 4.9) being sent from the source base station β_i to the respective neighbour base stations β_j where replica data of its local memory is being stored.

$$reqCommitment(\beta_i, \beta_j) = \{s_1, s_2, ..., s_n, ENC_{priv_{\beta_i}}(h(location))\} \tag{4.9}$$

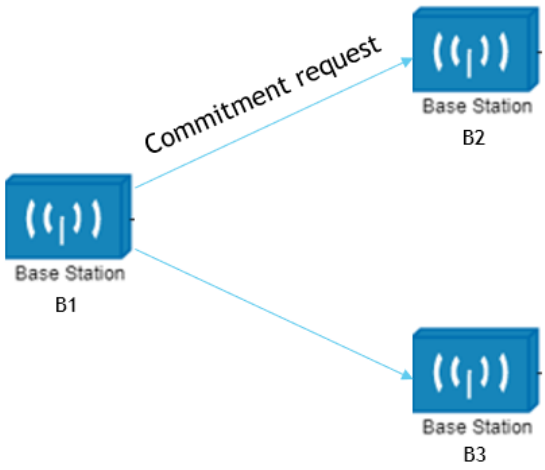

Figure 4.9: Commitment request from source base station to neighbour base station

The reply message from the destination base station should contain all the sensor node data hashes along with their respective logical clock values and a signature generated by the destination base station by encrypting the obtained source location using its private key. Appending logical clock and sending this value helps the source base station to check the data only until that value in its local memory. It also helps in maintaining data freshness and synchronization of data among base stations as they will receive the up to date logical clock value of replica base stations. This helps the source base station to find missing data and to send the newer data if present to replica base station. Fig. 4.10 shows destination base station sending data to source base station as a commitment response (Eq. 4.10).

$$resCommitment(\beta_j, \beta_i) = \{h(s_1), lc(s_1), ..., h(s_n), lc(s_n), ENC_{priv_{\beta_j}}(h(location))\} \tag{4.10}$$

This commitment response from the destination or neighbour base station (β_j) helps source base station (β_i) to calculate the same hash of every sensor node in the local memory using the obtained logical clock. The source base station also verifies if the signature from the destination base station matches, this helps in achieving integrity. The fig. 4.11 shows source base station verifying the replica data.

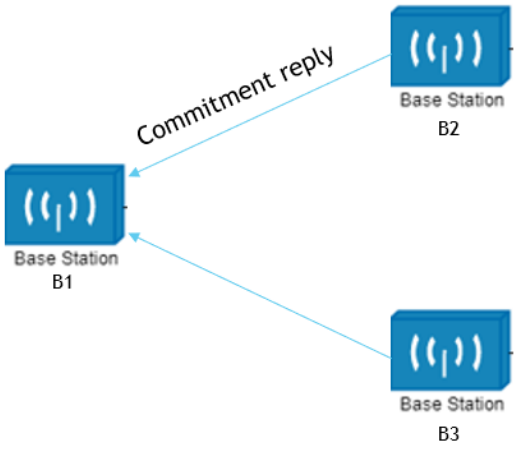

Figure 4.10: Commitment response from neighbour base station to source base station

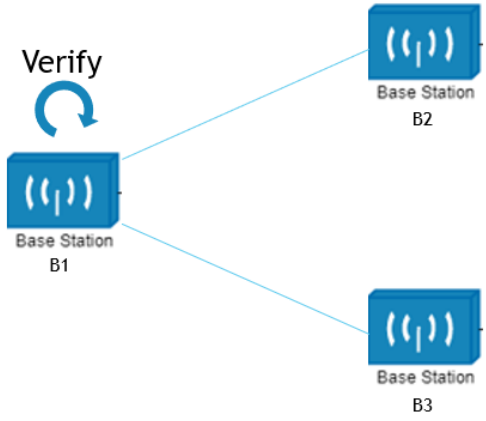

Figure 4.11: Commitment verification done by source base station

Chapter 5

Evaluation Methodology and Analysis

This chapter evaluates the various components of the SEA framework employing different methodologies, such as computer emulation, comparison with existing approach, and formal security analysis. We have used Python based framework for the computer emulation, and Scyther protocol analyzer for the formal verification of the secured phases. The following section highlights these methodologies in details.

5.1 Evaluating Availability of Data

5.1.1 Computer Emulation

Platform Setup

The platform used to simulate the sensor data movement among sensor nodes, base stations is called Spyder Integrated Development Environment. It is an open source cross-platform integrated development environment for scientific programming in the Python programming language bundled with numerous Python packages such as NumPy, SciPy, Pandas, IPython, Matplotlib etc. Spyder IDE is from the Anaconda Scientific Python Distribution Environment. This IDE is installed on windows 10 operating system with 12 GB of RAM having 2 physical cores and 4 logical processors.

Programming Framework

The memory allocation and replication concepts are evaluated by the generated CSV files and text files, which are created using python programming. A random 8-bit binary numbers data file is generated in order to resemble 8 bit sensor data. This data is generated using the random library of Python, which produces random numbers based on the specified seed. A python program "8BitNumGen.py" creates a file "Random8BitFile.txt" that includes 1 billion records. This file is used as a source of input data for the sensor nodes. The data in the file are randomly generated

and used by the main program "SEA.py". Apart from the data input, the main program requires 7 different input parameters for the dynamic wireless sensor network configuration setup. The parameters are defined as follows:

1. Number of sensor nodes (x): Based on the input value for x number of sensor nodes, x number of variables are created in the program.

2. Number of base stations (n): Number of CSV files generated depends on the number of base stations input.

3. Replication factor (r): This is used to create replicas randomly between base stations using random choice function from random library.

4. Sensing Time: Sensing time is the number of times every sensor should sense data.

5. Configuration Change Time: The input from this is used to change the complete configuration of the network. This is used to resemble the dynamic movement of sensor nodes and base stations across the network.

6. Sensing Frequency: This input is in seconds, which is used by the sensors to understand how often they need to sense data in the network.

7. Base Station Storage Capacity: This is used to calculate the free memory (mem^{β}_{free}) and keep track of occupied memory ($mem^{\beta}_{occupied}$).

In this phase of configuration, various sensor nodes are randomly assigned to one base station. Base stations are also randomly connected in this emulator using the imported random library. The replication factor determines the number of connections among base stations. Replication factor r indicate $r - 1$ number of random connections to the other base stations. After the configuration of the network is set up, sensors start sensing data randomly from the Random8BitFile.txt. Data from the sensor nodes is sent to its connected base station. Based on the other connected base stations, incoming data will be replicated by the source base station to the connected destination base stations. All the data from sensor nodes, base stations local memory (mem^{β}_{local}) and remote memory (mem^{β}_{remote}) along with their connections are recorded

using CSV and text files. The common files generated for any network configuration include config.csv, memory.csv, neighbor.txt, sensordata.csv, topology.txt, trigger.csv and other m number of base station CSV files.

Fig. 5.1 highlights the working principle of the program execution. The program execution starts with the prompt for inputs to configure and design a network with a set of sensor nodes and base stations. Based on the provided replication factor and other inputs, connections between base station and neighbor selection takes place. This is then followed by random selection of sensor data into the sensor nodes and sending this data to its connected base stations. The next step allows the data replication based on the neighbour connection setup. After this process, the program waits for some time based on the provided sensing frequency. This process is again repeated until base station is about to overflow and then the base station flushes its local or remote memory based on the amount of data it is stored for every round. This memory freeing is also reflected in the other neighbouring sensor nodes. After this check, base station also checks if it has reached the configuration change time and according changes its configuration by flushing all its memory. This entire process comes to a stop when the current time reaches the provided sensing time.

Output File Description

Configuration File

This file describes the configuration of the network at different times. It has information of the network such as number of base stations (n) present in the network, number of sensor nodes present in the network, base station memory in bits (mem_β), sensor memory in bits (mem_s), replication factor (r) in the network (Tab. 5.1).

Table 5.1: Configuration file format

Time	No. of Base stations (n)	No. of sensors (x)	mem_β	mem_s	r

Memory File

It has all the information of base station memory along with the ID of the base station (ID_β) at time t_i. Tab. 5.2 shows the amount of memory free (mem^β_{free}) and the amount of memory occupied ($mem^\beta_{occupied}$) with sensor data.

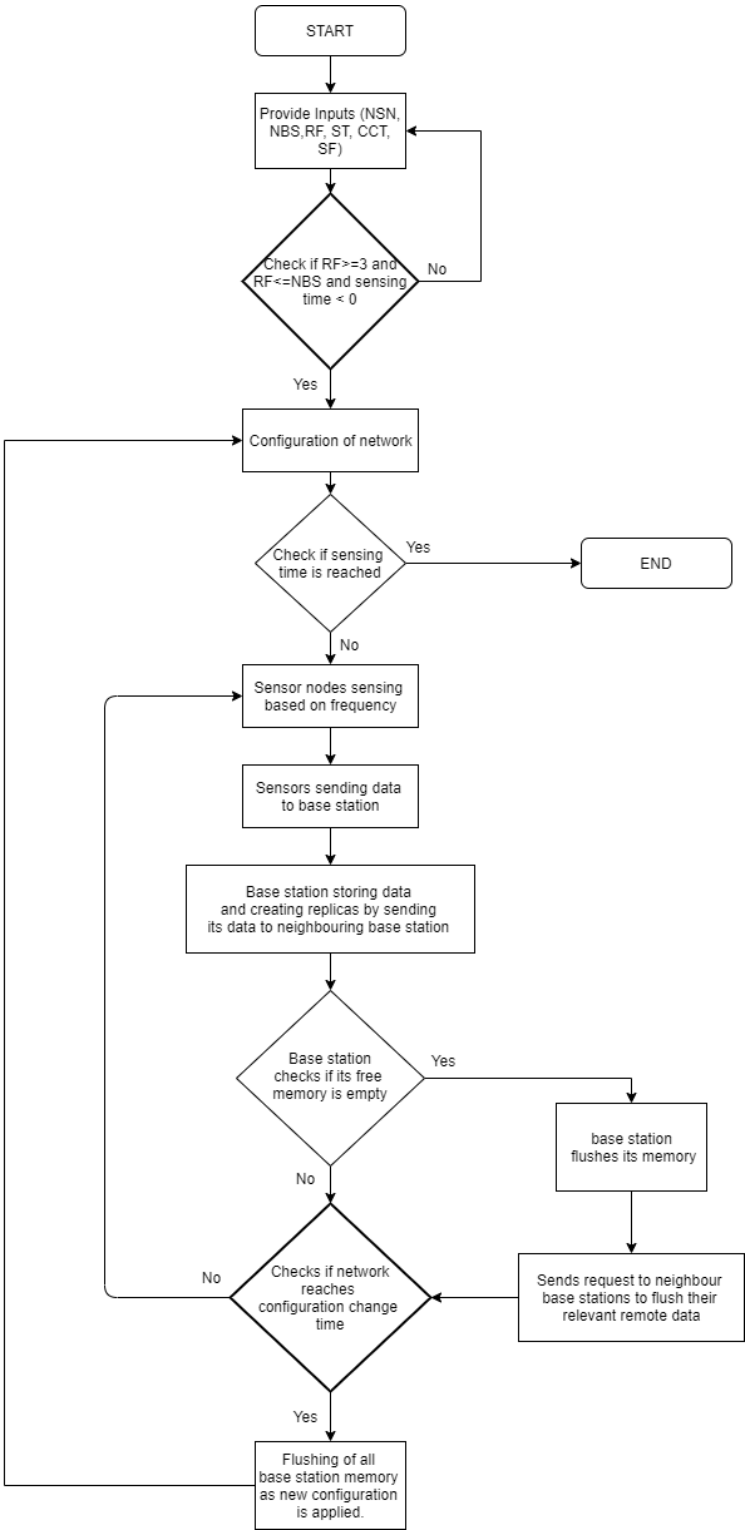

Figure 5.1: The flow chart shows how the programming framework executes its logic

Table 5.2: Memory file format

t_i	ID_β	mem^{β}_{free}	$mem^{\beta}_{occupied}$

Neighbour Information

It has information on the number of neighbors connected to every base station at different times (Tab. 5.3). It also shows all the ID's of connected neighboring base stations to the source base station at time t_i.

Table 5.3: Neighbour file format

t_i	ID_β	No. of Neighbours (k)	Neighbours (β_k)

Sensor Data

This CSV file (Tab. 5.4) has information on the number of sensor nodes connected to a base station along with all the data the sensor nodes have sensed at a different time t_i.

Table 5.4: Sensor data file format

t_i	sensor IDs (ID_s)	sensor readings ($data_s$)

Local and Remote Sensors

The Topology.txt file gives overall view of the sensor nodes connected to a base station (Tab. 5.5). It provides information about directly connected sensor nodes to the base station which represents what local memory is being stored. It also shows the remote sensors data being stored in its memory at different time t_i.

Table 5.5: Format of Topology.txt file

t_i	ID_β	Local Sensors	Remote Sensors

Memory Flush

This file serves as the log file showing all the information of every base station at different times, when ever data is being flushed due to configuration change or when its memory is used up (Tab. 5.6).

Table 5.6: The log file provides information on Memory flush with Base Station ID

t_i	Base Station IDs (ID_β)	Flush Type

Flushing or offloading the base station data to the server or user is done when the configuration changes or when the base station memory limitations are met. The concept of flushing involves base stations to send all the data to the server and erase its existing memory to free up space for new sensor data. The proposed protocol uses two different flushing mechanisms, local and complete flush which offloads either local memory or complete memory of the base station respectively.

Based on computer simulation, we check how efficiently memory allocation and deallocation will be performed in order to minimize the over and under-utilization of memory. Memory requirements and storage capacity of base stations differ in real world scenarios, the proposed approach assumes the storage capacity of base station to be 256 bits. Furthermore, this simulation provides a way of replicating real world scenarios by introducing forced delay in the programming framework.

Configuration for experiment, in scenario 1:

Number of Sensor Nodes (x): 9

Number of Base Stations (n): 3

Replication Factor (r): 3

Sensing Time (ST): 360 times

Configuration Change Time (CCT): 120 times

Sensing Frequency (SF): sense every 10 seconds

Sample outputs:

Config.csv: Tab. 5.7 configuration file is a sample snapshot of scenario 1 that shows the configuration of deployed sensor network at time t0 to t15. It contains information about number of base stations, number of sensor nodes, base station memory or storage capacity, sensor memory and Replication factor. This file is used to identify the status of the network. Any change in the network configuration is first reflected in this file i.e., compromising of nodes, addition of new nodes is periodically updated in this configuration file. The file is updated with new data when a round of sensing, base station storing and replication is completed at every time t_i.

Table 5.7: Config CSV file entry based on scenario 1

T	NumBS	NumSN	BaseStationMemory	SensorMemory	ReplicationFactor
t0	3	9	256	8	3
t1	3	9	256	8	3
t2	3	9	256	8	3
t3	3	9	256	8	3
t4	3	9	256	8	3
t5	3	9	256	8	3
t6	3	9	256	8	3
t7	3	9	256	8	3
t8	3	9	256	8	3
t9	3	9	256	8	3
t10	3	9	256	8	3
t11	3	9	256	8	3
t12	3	9	256	8	3
t13	3	9	256	8	3
t14	3	9	256	8	3
t15	3	9	256	8	3

Memory.csv: The memory.csv file shows amount of memory used and the amount of free memory present in every base station periodically. Tab. 5.8 is a sample snapshot of the memory.csv file from time t0 to t6. The data of each base station along with its free memory is monitored. If the base station reaches a point where the free memory is less than base stations associated sensor nodes memory, it does a whole flush of its data and sends this information to neighboring base stations. If the base stations memory is less than its directly connected sensor nodes plus remotely connected sensor nodes memory, it flushes the local data in its memory and sends a request to the neighboring replica base stations. In this configuration, the occupied memory of each base station at time t0 is 72 and increases by 72 at time t1, making it 144, and at time t2 it is 216. However, the base stations can not accommodate more data with a configuration storage capacity of 256, since the data is increased by 72 bits each time, it flushes its entire memory to accommodate more memory by freeing up its storage capacity. This memory flushing helps the base station manage its memory periodically.

Fig. 5.2 highlights how a base station manages its memory. Occupied memory is captured based on the storage capacity of a base station. For every time t_i, a base station performs a memory check and depending on the situation, either executes a

Table 5.8: Memory CSV file entry based on scenario 1

T	BS ID	Occupied memory	Free Memory
t0	2	72	184
t0	1	72	184
t0	0	72	184
t1	2	144	112
t1	1	144	112
t1	0	144	112
t2	2	216	40
t2	1	216	40
t2	0	216	40
t3	2	72	184
t3	1	72	184
t3	0	72	184
t4	2	144	112
t4	1	144	112
t4	0	144	112
t5	2	216	40
t5	1	216	40
t5	0	216	40
t6	2	72	184
t6	1	72	184
t6	0	72	184

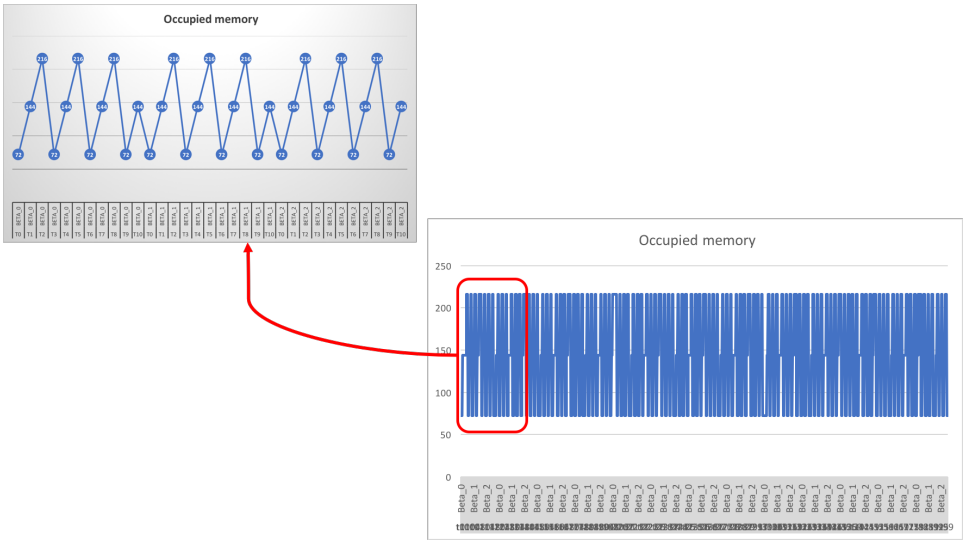

Figure 5.2: Occupied memory graph over time in SEA framework for scenario 1

local memory flushing or entire memory flushing. The type of flushing performed is logged in the trigger.csv for every flush. Base stations store and manage its replica

data in addition to managing its locally connected sensor data. If the storage capacity has reached its full capacity, it sends a request to the user if the memory is required. Based on the user response, the data will be sent and discarded from the base station storage or will be discarded directly. In the case of no response, data will be forwarded to the user and base station memory is flushed, thus preventing a memory overload crash of the base station and preserving availability. This allows base station to not communicate excessively with the end server, thus it reduces network usage and also minimizes the probability of man in the middle attacks. The proposed scheme functions better even with less memory resources and attempts to optimize the usage of existing memory automatically and because of this over-utilization or under-utilization of memory is avoided. This memory management and minimal replication in a distributed network enables data collection and processing in an effective manner by maximizing data availability and consistency across the network.

Neighbour.txt:

Neighbor.txt file has the information about base station ID, number of neighbors for every base station and the ids of neighboring base stations. The text file data provides information about 3 base stations β_0, β_1, β_2 and the number of connected neighboring base stations with IDs. According to the information in the text file , configuration change is not recorded in the sample data provided (Tab. 5.9). Thus connection between the base stations remain unchanged. The file provides information on neighbour data from time t0 to time t6.

Sensordata.csv: Sensordata.csv table shows the information about sensed data by different sensor nodes from time to time. (Tab.5.10) is a snapshot of the sensordata.csv file shows the sensed data by 7 sensor nodes s_0 to s_6 from time t0 to time t20. This file is used to validate if the base stations data being stored is actually the data being sensed by the sensor nodes.

Topology.txt

Topology.txt file has the information about base station ID and all the local sensors, and remote sensors data stored by the base station at regular intervals of time. Configuration changes can be tracked using this csv file, as we can see the change of the connected sensor nodes to the base station. The following sample data is the configuration of the base stations and the complete sensor nodeID'sstored in the network

Table 5.9: Sample neighbour file data

t_i	ID_β	No. of Neighbours (k)	Neighbours (β_k)

```
t0, 2, 2, [1, 0]
t0, 1, 2, [2, 0]
t0, 0, 2, [2, 1]
t1, 2, 2, [1, 0]
t1, 1, 2, [2, 0]
t1, 0, 2, [2, 1]
t2, 2, 2, [1, 0]
t2, 1, 2, [2, 0]
t2, 0, 2, [2, 1]
t3, 2, 2, [1, 0]
t3, 1, 2, [2, 0]
t3, 0, 2, [2, 1]
t4, 2, 2, [1, 0]
t4, 1, 2, [2, 0]
t4, 0, 2, [2, 1]
t5, 2, 2, [1, 0]
t5, 1, 2, [2, 0]
t5, 0, 2, [2, 1]
t6, 2, 2, [1, 0]
t6, 1, 2, [2, 0]
t6, 0, 2, [2, 1]
```

from time t0 to t10. Base station β_0 stores sensor nodes s_1, s_6, s_7, s_0, s_4 in its local memory as they are directly connected and stores data of sensor nodes s_3, s_5, s_2, s_8 remotely, as these sensor nodes are being managed by neighboring base stations β_1 and β_2. Base station β_1 stores sensor nodes s_8 in its local memory as they are directly connected and stores data of sensor nodes s_3, s_5, s_2, s_1, s_6, s_7, s_0, s_4 remotely as these sensor nodes are being managed by neighboring base stations β_2 and β_0. Base station β_2 stores sensor nodes s_3, s_5, s_2 in its local memory as they are directly connected and stores data of sensor nodes s_8, s_1, s_6, s_7, s_0, s_4 remotely as these sensor nodes are being managed by neighboring base stations β_1 and β_0. Change of configuration introduces changes to the number of connected sensor nodes and connected sensor ID's, as the network is dynamic.

Fig. 5.3 shows how the topology is configured randomly using the emulator according to the given inputs during initial configuration. Bi-directional arrows denote

Table 5.10: Sample sensor data

T	0	1	2	3	4	5	6
t0	1010010	110	1111	101010	110111	10011110	11011110
t1	10110100	10101101	10100011	100111	10100100	1110110	111
t2	10100100	10001000	11110001	11010110	10000001	1100	1101000
t3	10111101	1001010	11111011	10010001	11111111	11011110	110011
t4	11111110	11000100	111011	10011101	110	10010011	1101110
t5	10101011	10011000	10010100	11010101	1000001	1101011	11010010
t6	11000111	11000000	11010101	10101	11111110	11001001	10001001
t7	10011110	11011	100101	10001111	111000	11001011	1111111
t8	1011101	11101110	10101	1011110	111110	1110010	110011
t9	10011000	10001110	10000	11110011	110010	101000	11100100
t10	10101101	1111000	1111011	1101000	1101110	11000101	11100101
t11	11110100	1111001	10010	11100110	1101100	11000010	111110
t12	100100	11100010	10010	11010	11101010	1001101	1111101
t13	10110011	1011010	10001000	10001010	10000001	10100	101110
t14	1010101	1011100	10001101	10011001	111111	1110001	111110
t15	111	1001011	11011110	10010010	1000110	10111111	10101010
t16	101011	1011010	10111011	1101001	11011111	11100010	10100110
t17	1111110	11101101	11011100	11010111	1101100	1000110	10101100
t18	10101100	111000	1010010	10011010	10011011	10100101	10100111
t19	11000111	110101	1100110	11011	10101011	1011	11011111
t20	1011101	10001110	10100101	1010001	11101010	1101	11110000

two-way communication between the nodes, unidirectional arrow denotes information is being sent from the one node to the pointing node. This configuration has no one-way arrows, as every base station is connected to each other as a mesh network.

Trigger.csv

Trigger.csv is a log file having information about flushing of memory. For every flush at time tn, data in this log file is updated. To understand memory management in this proposed scheme and match the memory.csv file, we use trigger.csv file to check what type of flush has occurred. Sample data in (Tab. 5.12) shows logs from time t0 to t21. Local flush has not occurred according to this configuration and data being sent and stored.

Configuration for experiment, in scenario 2:

Number of Sensor Nodes (x): 15

Number of Base Stations (n): 5

Replication Factor (r): 4

Table 5.11: Sample data of Topology.txt file

```
T, BS ID, Local_Sensors, Remote_Sensors
t0, 0, [1, 6, 7, 0, 4], [3, 5, 2, 8]
t0, 1, [8], [3, 5, 2, 1, 6, 7, 0, 4]
t0, 2, [3, 5, 2], [8, 1, 6, 7, 0, 4]
t1, 0, [1, 6, 7, 0, 4], [3, 5, 2, 8]
t1, 1, [8], [3, 5, 2, 1, 6, 7, 0, 4]
t1, 2, [3, 5, 2], [8, 1, 6, 7, 0, 4]
t2, 0, [1, 6, 7, 0, 4], [3, 5, 2, 8]
t2, 1, [8], [3, 5, 2, 1, 6, 7, 0, 4]
t2, 2, [3, 5, 2], [8, 1, 6, 7, 0, 4]
t3, 0, [1, 6, 7, 0, 4], [3, 5, 2, 8]
t3, 1, [8], [3, 5, 2, 1, 6, 7, 0, 4]
t3, 2, [3, 5, 2], [8, 1, 6, 7, 0, 4]
t4, 0, [1, 6, 7, 0, 4], [3, 5, 2, 8]
t4, 1, [8], [3, 5, 2, 1, 6, 7, 0, 4]
t4, 2, [3, 5, 2], [8, 1, 6, 7, 0, 4]
t5, 0, [1, 6, 7, 0, 4], [3, 5, 2, 8]
t5, 1, [8], [3, 5, 2, 1, 6, 7, 0, 4]
t5, 2, [3, 5, 2], [8, 1, 6, 7, 0, 4]
t6, 0, [1, 6, 7, 0, 4], [3, 5, 2, 8]
t6, 1, [8], [3, 5, 2, 1, 6, 7, 0, 4]
t6, 2, [3, 5, 2], [8, 1, 6, 7, 0, 4]
t7, 0, [1, 6, 7, 0, 4], [3, 5, 2, 8]
t7, 1, [8], [3, 5, 2, 1, 6, 7, 0, 4]
t7, 2, [3, 5, 2], [8, 1, 6, 7, 0, 4]
t8, 0, [1, 6, 7, 0, 4], [3, 5, 2, 8]
t8, 1, [8], [3, 5, 2, 1, 6, 7, 0, 4]
t8, 2, [3, 5, 2], [8, 1, 6, 7, 0, 4]
t9, 0, [1, 6, 7, 0, 4], [3, 5, 2, 8]
t9, 1, [8], [3, 5, 2, 1, 6, 7, 0, 4]
t9, 2, [3, 5, 2], [8, 1, 6, 7, 0, 4]
t10, 0, [1, 6, 7, 0, 4], [3, 5, 2, 8]
t10, 1, [8], [3, 5, 2, 1, 6, 7, 0, 4]
t10, 2, [3, 5, 2], [8, 1, 6, 7, 0, 4]
```

Sensing Time (ST): 360 times

Configuration Change Time (CCT): 120 times

Sensing Frequency (SF): sense every 10 seconds

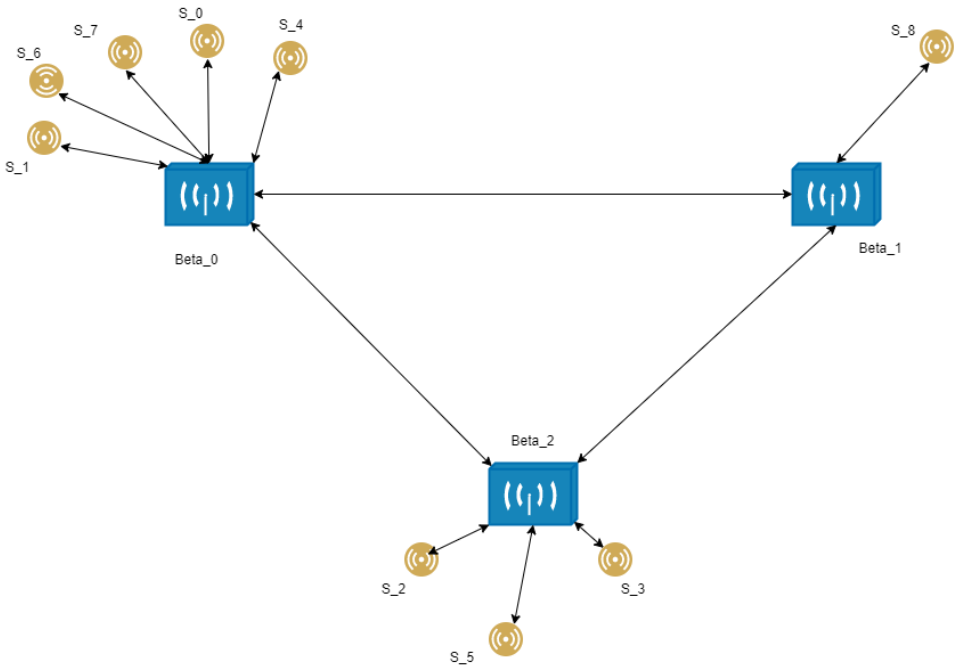

Figure 5.3: Randomly configured topology

Outputs:

The memory.csv file displays the amount of memory used and the amount of free memory available in each base station from time to time. The table in Appendix - Scenario 2 is a sample snapshot of the memory.csv file from time t0 to t5. Each base stations data along with its free memory is tracked. If the base station reaches a point where the available memory is less than base stations connected sensor nodes memory, it does a whole flush of its data and sends the information to neighboring base stations. If the usable base station memory is less than its directly connected sensor nodes and remotely connected sensor nodes memory, it flushes local data fromits memory and sends a request to the neighboring replica base stations. In this configuration, base station β_0 gets 112 bits of data for every round, base station β_1 gets 96 bits of data, base station β_2, β_3 and β_4 gets 88, 160, 104 bits of data respectively. At a certain point, base stations can not store additional data with a hardware storage capacity of 256 bits, since the data is increased by more than 88 bits each second, base station flushes its whole memory to avoid memory overload to handle more data by freeing up the storage space. This flushing of memory helps the base station manage memory its memory from time to time.

Table 5.12: Sample log file

T	BS ID	
t3	0	complete data flushed
t3	1	complete data flushed
t3	2	complete data flushed
t6	0	complete data flushed
t6	1	complete data flushed
t6	2	complete data flushed
t9	0	complete data flushed
t9	1	complete data flushed
t9	2	complete data flushed
t12	0	complete data flushed
t12	1	complete data flushed
t12	2	complete data flushed
t15	0	complete data flushed
t15	1	complete data flushed
t15	2	complete data flushed
t18	0	complete data flushed
t18	1	complete data flushed
t18	2	complete data flushed
t21	0	complete data flushed
t21	1	complete data flushed
t21	2	complete data flushed

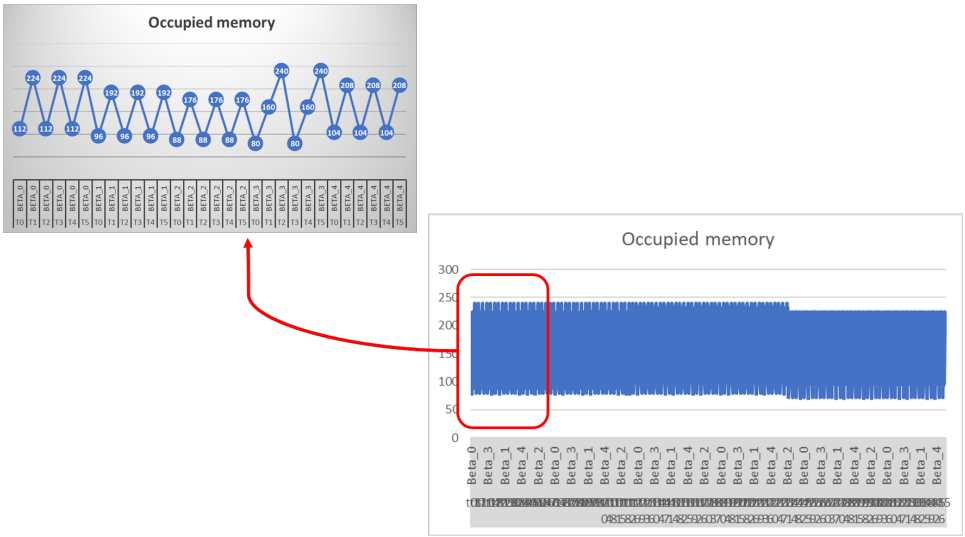

Figure 5.4: Graph shows occupied memory of base stations, where topology is designed based on scenario 2

Neighbour.txt file has information about base station ID, number of neighbors for every base station and the IDs of neighboring base stations. This sample neighbour file in Appendix - Scenario 2 provides information about 5 base stations β_0, β_1, β_2, β_3, β_4 and the number of connected neighboring base stations with IDs. According to the topology file information in Appendix - Scenario 3, configuration change is not recorded in the sample data provided. Thus connection between the base stations remain unchanged. This information provides neighbor data from time t0 to time t6. Sensordata.csv table shows the information about sensed data by different sensor nodes from time to time. Snapshot of the sensordata.csv file in Appendix - Scenario 2 shows the sensed data by 15 sensor nodes s_0 to s_14 from time t0 to time t20. This file is used to refer if the base stations data being stored is actually the data sensed from the sensor nodes.

Topology.txt file has the information about base station ID's along with all local sensors + remote sensors data stored by the base station from time to time. Changes in the configuration can be tracked using this csv file as we can see the change of connected sensor nodes to the base station. The sample data in Appendix - Scenario 2 is the configuration of base stations and complete sensor data IDs being stored in the network from time t0 to t6. Base station β_0 stores sensor nodes s_6, s_{13}, s_1 in its local memory as they are directly connected and stores data of sensor nodes $s_9, s_4, s_{14}, s_5, s_7, s_{10}, s_0, s_2, s_8, s_3, s_{12}$ remotely as these sensor nodes are being managed by neighboring base stations β_1, β_3 and β_4. Base station β_1 stores sensor nodes s_9, s_4, s_{14}, s_5 in its local memory as they are directly connected and stores data of sensor nodes $s_7, s_{10}, s_0, s_2, s_8, s_3, s_{12}, s_{11}$ remotely as these sensor nodes are being managed by neighboring base stations β_3, β_4 and β_2. Base station β_2 stores sensor nodes s_{11} in its local memory as they are directly connected and stores data of sensor nodes $s_6, s_{13}, s_1, s_7, s_{10}, s_0, s_2, s_8, s_3, s_{12}$ remotely as these sensor nodes are being managed by neighboring base stations β_0, β_3 and β_4 and so on. However, base station β_2 which only stores s_{11} data is not being underutilized as it supports the network by storing replicas and thus increasing availability. Changes in configuration will cause variations in the number of connected sensor nodes and different sensor ID'S connected, as the network is dynamic.

Configuration for experiment, in scenario 3:

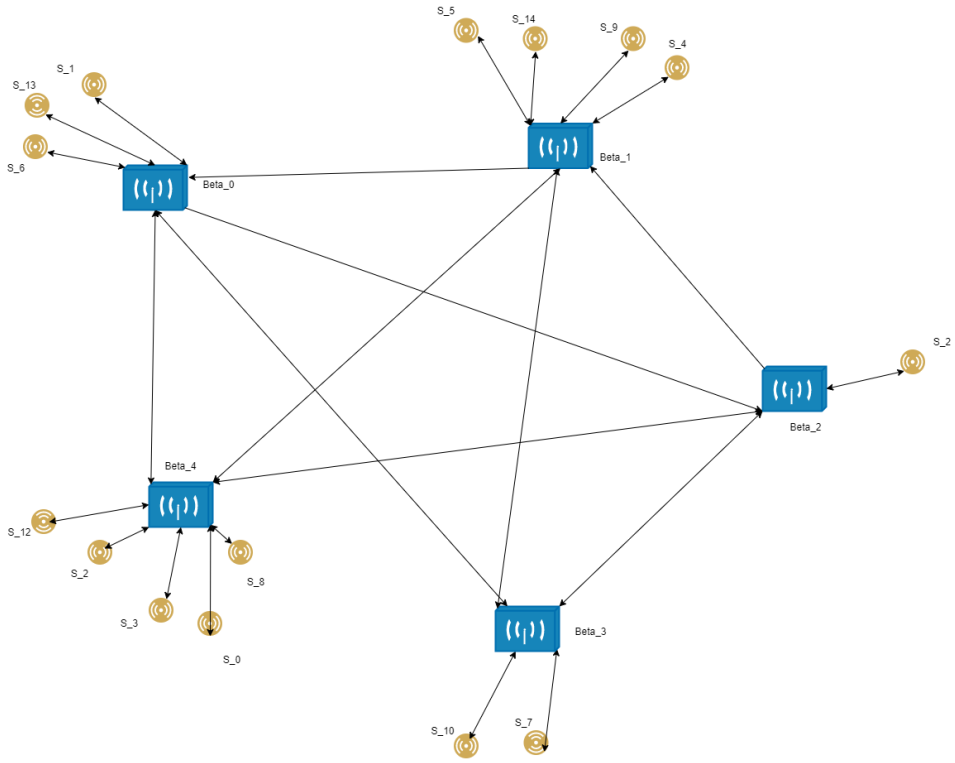

Figure 5.5: Topology of scenario 2

Number of Sensor Nodes (x): 30
Number of Base Stations (n): 10
Replication Factor (r): 4
Sensing Time (ST): 360 times
Configuration Change Time (CCT): 120 times
Sensing Frequency (SF): sense every 10 seconds
Outputs:

The table in Appendix - Scenario 3 is a sample snapshot of the memory.csv file from time t0 to t5. Each base stations data along with its free memory is monitored to avoid memory overload. If the base station reaches a point where the free memory is less than base stations connected sensor nodes memory, it does a whole flush of its data and sends the information to neighboring base stations. If the base stations memory is less than its directly connected sensor nodes + its remotely connected sensor nodes memory, it flushes its local data in its memory and sends a request to the neighboring

replica base stations. In this configuration, base station β_0 gets 96 bits of data for every round, base station β_1 gets 96 bits of data, base station $\beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8$ and β_9 gets 120, 88, 64,112,80,104,104 and 96 bits of data respectively. After a certain time, base stations cannot accommodate more data having its storage capacity at 256 bits (according to the configuration). As the data is increased by more than 80 bits every time except for base station β_4 and β_6,these base stations flushes its complete memory before the overflow after sensing at time t2 to accommodate more memory and free its storage capacity. While base station β_6 flushes its memory after time t3 since it only adds 80 bits of memory each time and β_4 flushes after time t4 as it only adds 64 bits of memory each time respectively. This flushing of memory helps the base station manage its memory periodically. (Fig. 5.6) demonstrates how base stations handle memory from time to time and control its capacity by saving and flushing data as necessary. This strategy allows base stations support each other and exchange data with each other to prevent a single point of failure.

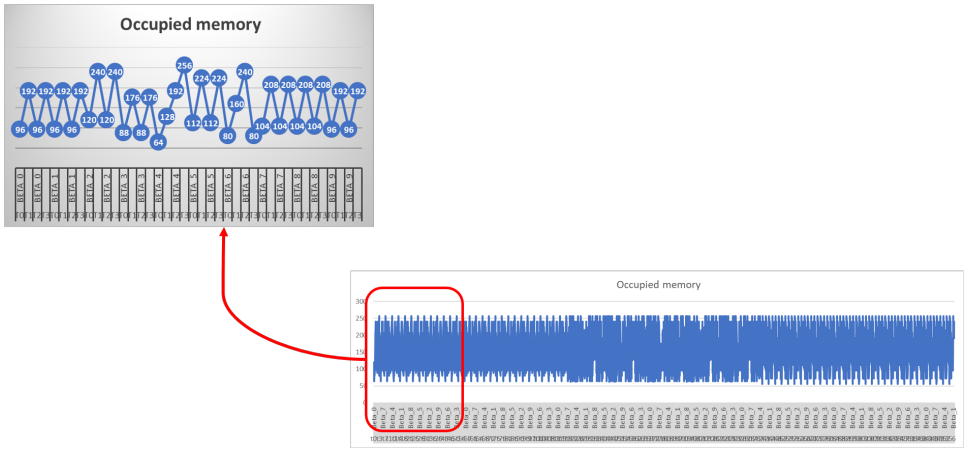

Figure 5.6: Topology of scenario 3

Neighbor.txt file has the information about base station ID, number of neighbors for every base station and the IDs of neighboring base stations. The neighbour file data in Appendix - Scenario 3 provides information about 10 base stations $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9$ and the number of connected neighboring base stations with IDs. According to the information in Appendix - Scenario 3 file, configuration change is not recorded in the sample data provided. Thus, connection between the base stations remain unchanged. The file provides neighbor data from time t0 to

time t3. Sensordata.csv table shows the information about sensed data by different sensor nodes. Snapshot of the Sensordata.csv file in Appendix - Scenario 3 shows the sensed data by 30 sensor nodes s_0 to s_{29} from time t0 to time t20. This file is used to refer if the data being stored in base stations is actually the data sensed from the sensor nodes. Topology.txt file in Appendix - Scenario 3 has the information about base station ID and all the local sensors + remote sensors data stored by the base station from time to time. Changes in the configuration can be tracked using this csv file as we can see the change of connected sensor nodes to the base station. The sample data in Appendix - Scenario 3 is the configuration of base stations and complete sensor data IDs being stored in the network from time t0 to t6. Base station β_0 stores sensor nodes s_2, s_{17}, s_8 in its local memory as they are directly connected and stores data of sensor nodes$s_7, s_1, s_{27}, s_{15}, s_9, s_3, s_{24}, s_{11}, s_4$remotely as these sensor nodes are being managed by neighboring base stations β_9, β_4 and β_5. Base station β_1 stores sensor nodes s_{12} in its local memory as they are directly connected and stores data of sensor nodes $s_{25}, s_{18}, s_{28}, s_{20}, s_7, s_1, s_{27}, s_{15}, s_2, s_{17}, s_8$ remotely as these sensor nodes are being managed by neighboring base stations β_2, β_9 and β_0. Base station β_2 stores sensor nodes $s_{25}, s_{18}, s_{28}, s_{20}$ in its local memory as they are directly connected and stores data of sensor nodes $s_5, s_6, s_{19}, s_{13}, s_{14}, s_{22}, s_{26}, s_{21}, s_{24}, s_{11}, s_4$ remotely as these sensor nodes are being managed by neighboring base stations β_8, β_6 and β_5 and so on. This makes sure that, when base stations have less connected sensor nodes, it will help the network by storing remote data from neighboring base stations. Base station β_1, β_3 only stores one sensor node data in its local memory, however it stores more than 10 other sensor node replicas data, hence base station's β_1, β_3 storage is not underutilized, thus increasing availability. Change of configuration can bring changes in the number of sensor nodes connected, and sensor ID's connected as the network is dynamic.

Memory limitation of base station varies from hundreds of kilobytes to megabytes. The proposed approach considers 256 bits as the storage capacity to show the memory management, data flushing of the proposed approach and also to compare the performance with blockchain. In real world scenario, the framework is believed to perform even better due to larger storage capacity of base stations which requires far fewer

flushing with the proposed approach, whereas base stations using blockchain technology replication consume memory frequently due to higher replication factor leading to over utilization of storage limits. In addition, the sensor data size varies in real world scenarios from 20 bits or more which includes timestamp, actual sensed data value etc. The storage capacity could be increased from 256 bits to a larger number in the emulator, if required, and it will not have any negative impact on the proposed framework in terms of security. Similarly, the sample size of the sensor nodes data does not have any undesirable effect on the proposed system, it still guarantees the efficiency and promises security as with 8 bit sensor data.

5.1.2 Comparison of Memory Management

SEA Memory Management

Configuration for experiment, in scenario 1:

Number of Sensor Nodes (x): 9
Number of Base Stations (n): 3
Replication Factor (r): 3
Sensing Time (ST): 360 times
Configuration Change Time (CCT): 120 times
Sensing Frequency (SF): sense every 10 seconds

A dynamic wireless sensor network having 3 base stations and 9 sensor nodes in this scheme will have 3 replicas in total. This is because our approach considers 3 number of replicas as ideal and efficient replication factor [19] to increase availability of data in the network with minimal storage needs. Even in the case of adversaries or attacks in the network, chances of data loss or compromising is less. This scenario generates data of about 72 bits for every 10 seconds due to the sensing frequency. As every base station stores all the 9 sensors data, running this network for 10 minutes produce 540 bytes of sensor data with 1620 bytes of total data including duplicates. One hour of running this network will generate 9720 bytes of total data considering this replication factor. At each base station, 3240 bytes of data is stored.

Configuration for experiment, in scenario 2:

Number of Sensor Nodes (x): 10

Number of Base Stations (n): 5

Replication Factor (r): 3

Sensing Time (ST): 360 times

Configuration Change Time (CCT): 120 times

Sensing Frequency (SF): sense every 10 seconds

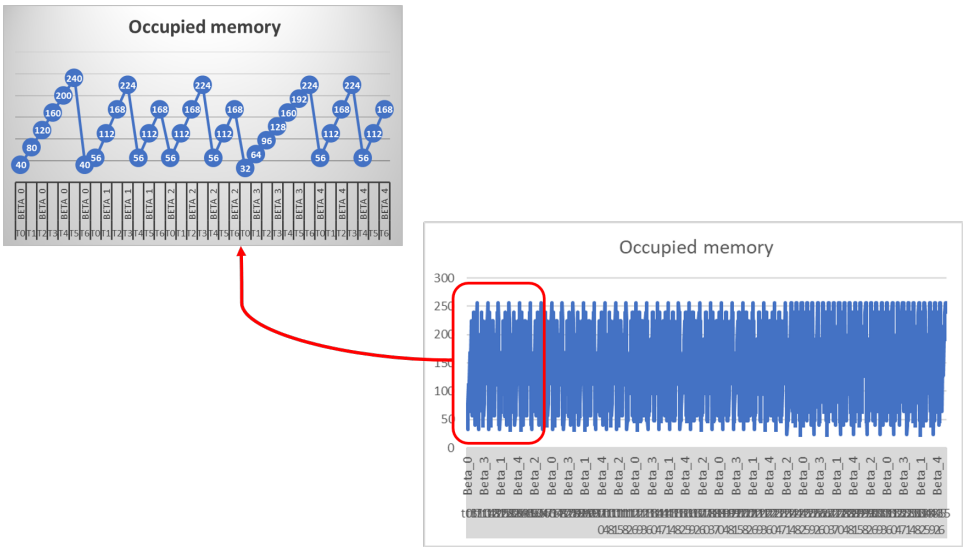

Figure 5.7: Occupied memory graph over time in SEA framework for scenario 2

According to our approach, three replicas are considered for this configuration with 10 sensor nodes and 5 base stations. This network generates 80 bits of data for every round (i.e., 10 seconds as per the sensing frequency). The number of base stations in this configuration is not equal to the replication factor, that causes base stations not to have all the network data present in their storage. As per the figure above, base station β_0 stores 40 bits of data every 10 seconds with 0 bytes of data in its local memory, as no sensor nodes are directly connected to it, however it helps the network by storing 40 bits of replica data every 10 seconds in its remote memory. Base stations β_1, β_2, β_3, β_4 stores 56, 56, 32 and 56 bits of data respectively in its memory for every 10 seconds. According to the amount of data being stored for every round, base station keeps track of storage overflow by releasing its memory when required. This configuration allows the presence of replicas in the network providing availability with minimal duplication of data. This also helps in the reduction of network bandwidth usage and increases throughput, as every data is not replicated

more than required number of times. After 10 minutes of running this network, it generates a total of 1,800 bytes of memory. For one hour, 10800 bytes of data will be shared between the base stations.

Fig. 5.8 shows the topology of the network with 5 base stations and 10 sensor nodes. We also see that different sensor nodes are being managed by different base stations. The arrows denote the base stations connections to each other. $\beta_1 \rightarrow \beta_0$ denotes β_0 base station storing β_1 base stations data in its replica memory and so on. Bidirectional arrow represents two-way communication.

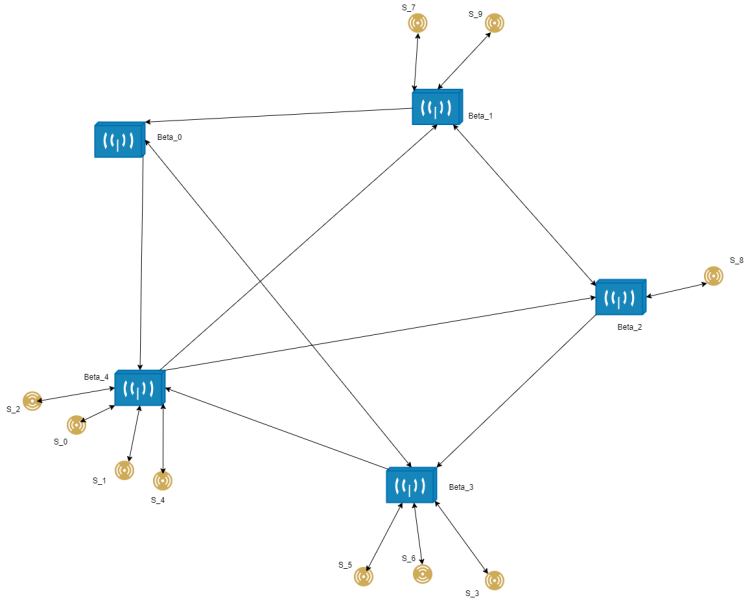

Figure 5.8: Topology of the network with 5 base stations and 10 sensor nodes

Evaluation in this research is in terms of memory overhead and communication overhead, which is optimized by considering the replication factor to a value which depends on the security requirements of the user for the network. Based on the network needs, replication factor can be variable. As it is configured before the network is deployed, network needs such as security and energy efficiency should be considered before the replication factor is determined. The higher the replication factor, the more communication overhead occurs, and the more energy is utilized, however data availability is equally increased in a similar fashion. The proposed approach considers 3 as the minimum replication factor value while the current literature in many other applications consider it to be an efficient and secure replication factor

value. Replication is considered to achieve data availability across the network. Regardless of the type of network either static or dynamic, if the network is considered to have sensitive and highly important information, deployed in unsecured and adversarial network conditions, requirements such as data availability and data Integrity are satisfied when the replication of data is considered in the network.

Blockchain Memory Management

Configuration for experiment, in scenario 1:

Number of Sensor Nodes (x): 9
Number of Base Stations (n): 3
Sensing Time (ST): 360 times
Configuration Change Time (CCT): 120 times
Sensing Frequency (SF): sense every 10 seconds

Replication in blockchain depends on the number of base stations present in the network. As per the above configuration, presence of 3 base stations mean 3 as the replication factor. This is similar to the proposed approach. The amount of data distributed and flushed remains the same with both the approaches in this configuration [39].

Configuration for experiment, in scenario 2:

Number of Sensor Nodes (x): 10
Number of Base Stations (n): 5
Sensing Time (ST): 360 times
Configuration Change Time (CCT): 120 times
Sensing Frequency (SF): sense every 10 seconds

Because of the existence of 5 base stations in the network, the replication factor would be 5 considering blockchain for this setup. Increasing the number of replicas would produce more duplicates in the network, resulting in increased network usage, increased communication overhead, memory overhead, increased required computational power and power consumption. As this network generates 80 bits of data for every round i.e., 10 seconds of sensing frequency, amount of data stored in the base stations vary by a large number. The figure above shows that each base station stores

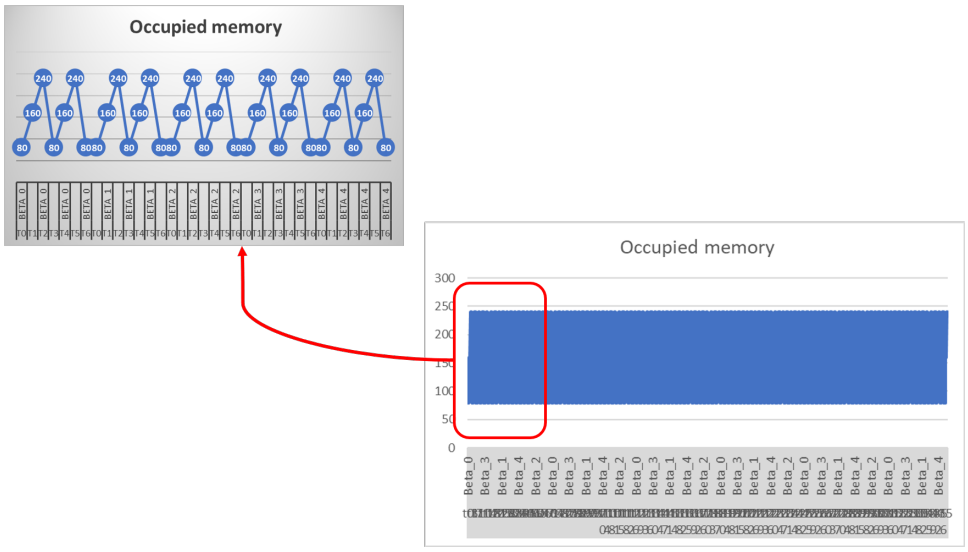

Figure 5.9: Occupied memory graph over time in Blockchain framework for scenario 2

80 bits of data every 10 seconds. For 10 min of this network execution, base stations store 3000 bytes of data altogether. 1 hour network run produces nearly 18000 bytes of data to be exchanged between base stations. In the next section, analysis of scenario 2 will be explained in detail.

Fig. 5.10 shows the topology of the network with 5 base stations and 10 sensor nodes. We can also see various sensor nodes being managed by different base stations. The arrows denote the base stations connected to each other. Bidirectional arrow represents two-way communication.

Comparison using output files in detail: Considering scenario 2: The amount of total memory stored in SEA for 1 min is 180 bytes, whereas network implemented using blockchain produces 300 bytes of data. After 10 minutes of network operation, the data stored in SEA is 1800 bytes compared to 3000 bytes produced using blockchain. With the increase in the amount of time to 1 hour, SEA needs to handle 10800 bytes of data compared to blockchain dealing with 18000 bytes of data. As time progresses, the amount of data generated by blockchain varies greatly in comparison with SEA.

Regarding the number of data flushes needed for the two methods, each base station in SEA flushes its memory 12.4 times in average for one minute; while blockchain base stations have to flush their data twenty times a minute. This difference is even

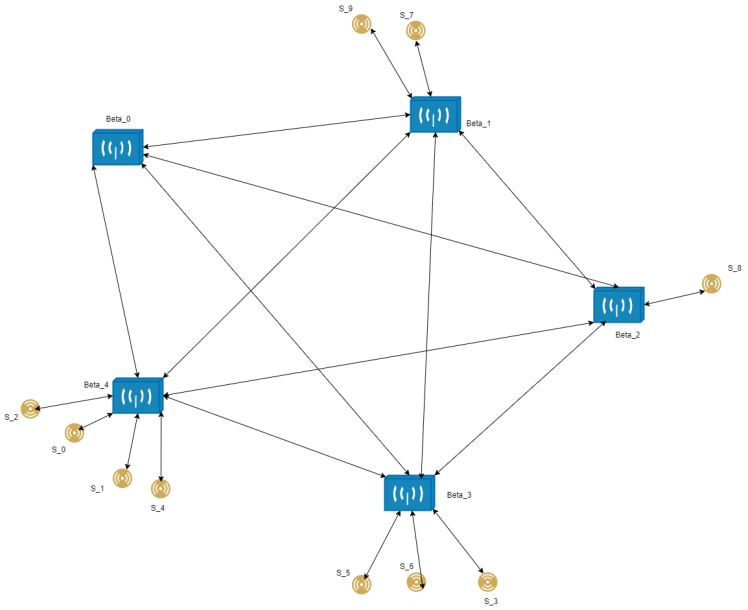

Figure 5.10: Topology of the network with 5 base stations and 10 sensor nodes

greater when compared to one hour of network operation, as each base station in SEA flushes its data 71 times compared to blockchain base stations flushing its data 117 times. This comparison is conducted using SEA and blockchain configuration variants provided by trigger.csv files.

The sensed data transfer from each neighbouring base station is determined by means of the generated topology.txt file for different times. Detailed information based on the comparison is presented in (Tab. 5.13):

In contrast with the proposed solution, communication between base stations in blockchain increases by a huge amount. In addition, the configuration used to compare and simulate the scenario has fewer sensor nodes and less base stations, we assume that real-time scenarios display much greater data distribution and delivery efficiency using SEA, helping to minimise energy usage and improve network lifetime.

5.2 Security Evaluation

5.2.1 Formal Security Analysis

We evaluate the proposed security schemes using Scyther tool. Scyther is used to automatically verify security protocols and analyze different classes of adversary models or vulnerabilities present in the framework. The number of runs is configured in this

Table 5.13: Comparison of data transmission between SEA and Blockchain

Time	Base Stations	SEA	Blockchain
1 Minute	β_0	30 bytes	60 bytes
	β_1	30 bytes	48 bytes
	β_2	36 bytes	54 bytes
	β_3	6 bytes	42 bytes
	β_4	18 bytes	36 bytes
10 Minutes	β_0	300 bytes	600 bytes
	β_1	300 bytes	480 bytes
	β_2	360 bytes	540 bytes
	β_3	60 bytes	420 bytes
	β_4	180 bytes	360 bytes
1 Hour	β_0	1.7578 KB	3.5156 KB
	β_1	1.7578 KB	2.8125 KB
	β_2	2.1093 KB	3.1640 KB
	β_3	0.3515 KB	2.4609 KB
	β_4	1.0546 KB	2.1093 KB

protocol analyzer according to the security scheme we are testing. For every run, Scyther is set to test different types of attacks possible with the proposed scheme in order to compromise the system. The proposed scheme mainly focuses on avoiding replay attacks and is validated to ensure the strongness of the system. Three different phases of the proposed framework are tested with this tool [21, 23].

Configuration Validation

The proposed configuration phase uses public key encryption and signing mechanism for the purpose of key distribution and neighbor selection. It also involves a simple hashing algorithm to generate hash values of data to ensure integrity of the message and for authentication. Every base station will send a connection request to all the base stations with a message containing a randomly generated symmetric key, signing this key using the private key from generated asymmetric cryptography key pair and hash of its location. This message is encrypted by the public key of destination base station to maintain secrecy of the message. This is followed by a response message to confirm the connection from the destination which has symmetric key XORed with $h(location)$ signed by the private key of destination base station and encrypted using

Table 5.14: Security Analysis of the SEA Framework using Scyther during Configuration Phase

Role	Sl.Num.	Claim	Status	Comment
β_i	1	secret $sk_{\beta_i\beta_j}$	Ok	No Attacks
	2	secret $\delta t_{diff}^{\beta_j}$	Ok	No Attacks
	3	secret $location$	Ok	No Attacks
	4	secret $\{(location) \oplus sk_{\beta_i\beta_j}\}$	Ok	No Attacks
	5	Alive	Ok	No Attacks
	6	Weakagree	Ok	No Attacks
	7	Niagree	Ok	No Attacks
	8	Nisynch	Ok	No Attacks
β_j	1	secret $sk_{\beta_i\beta_j}$	Ok	No Attacks
	2	secret $\delta t_{diff}^{\beta_j}$	Ok	No Attacks
	3	secret $location$	Ok	No Attacks
	4	secret $\{(location) \oplus sk_{\beta_i\beta_j}\}$	Ok	No Attacks
	5	Alive	Ok	No Attacks
	6	Weakagree	Ok	No Attacks
	7	Niagree	Ok	No Attacks
	8	Nisynch	Ok	No Attacks

public key of source base station. The final message includes acknowledgement from the source base station, thus sending hash of its location along with the timediff for that destination base station. This is signed by its private key and encrypted using destinations public key to ensure confidentiality. As this configuration phase only occurs once, Scyther is set to only run for once and test using all types of attacks. We focused on analyzing the vulnerability and secrecy of δt_{diff}, $location$, symmetric key, $sk_{\beta_i\beta_j}$ and the XORed symmetric key and $h(location)$. Every parameter is verified to evaluated if it fails to establish a secure connection and avoid the presence of vulnerabilities. Tab. 5.14 outlines our arguments and the security of the scheme for several different cases. We also present detailed security assessment of claims verified using Scyther in this part.

Claim 1: δt_{diff} remains secret and confidential throughout the configuration process Timediff is mainly used by the source base station to decide and choose what base stations to connect. Base stations try to connect with other base stations having minimal timediff value. This value is also used for authentication purpose and key generation purpose in the key update phase. Timediff value is thus maintained secret in this process as it is encrypted using public key of the destination which can only

be decrypted by the private present with the destination base station.

Claim 2: location of source base station is fully encrypted and is kept secret Location of the base station stays the same until the end of the configuration. This value is hashed and sent to the other base stations. The source base station sends hash of location to destination base station which is used in the further process in commitments and key update phases after configuration. So this needs to be secret and its confidentiality is proved by Scyther tool.

Claim 3: hash function is kept secret Hash function is decided before dispersing the base stations into the network. So decision of hash function is not done in the communication, thus keeping it a secret.

Claim 4: XOR of location hash and symmetric key is kept confidential After destination base station receives symmetric key from the source base station. It confirms the connection by sensing a response containing symmetric key xored with hash of location. This xored message is signed by its prvate key and also encrypted using the source base stations public key. Thus keeping the message confidential and hidden.

Claim 5: Source base station and destination base stations remain alive during this configuration setup Both the base stations are believed to be alive in this configuration phase throughout the process. This claim is validated and verified using Scyther protocol analyzer tool. This protocol guarantees the aliveness of one base station to the other base station.

Claim 6: Ensures Weakagree The proposed protocol also guarantees the weak-agreement between base stations. The source base station is running the proposed scheme with the destination base station, likewise the destination base station is running the proposed scheme with the source base station. Communication is not affected by any adversary during the operation of the proposed scheme. This ensures and satisfies the weakagree claim. No adversary can initiate the start of the protocol by sending a request message.

Claim 7: Guarantees Niagree between source and destination base station Ni-agree or non-injective agreement implies that during the activity of the proposed scheme, the source and the destination base station agree to any data exchange. The source base station can securely transmit information to the destinationbase station

during the operation of the proposed scheme and vice-versa. If Niagree fails, then we can infer that a man-in-the-middle attack occurs during protocol execution. Our argument, however, is evaluated using Scyther protocol analyzer.

Claim 8: Maintains Nisynch throughout this process Nisynch or non-injective synchronization is true if all steps before the claim are carried out in accordance with the proposed scheme definition. Nisynch ensures that during the authentication process, no synchronization issue occurs between source and destination base station. Nisynch further demands that the occurrence ofeach message isfollowed by the event that the message was sent. Nisynch in the proposed scheme is validated using the Scyther protocol analyzer.

Key Update Validation

Key update phase in this approach helps both the base station and destination base stations to update the previously setup symmetric key without any transmission of new key between them. Source base station sends the key update request to which destination base station sends a response to acknowledge this request confirmation. The message from source base station contains hash of its location and timediff, xored together, signed by the source and encrypted using the public key of neighboring base station. This is followed by the response message containing hash of source base stations location signed by the neighboring base stations private key and encrypted using source base stations public key. This response message includes location to make authenticate the neighboring base station as well as to act as the confirmation of key update request. This is tested using many attacks and is run for 5 times using the Scyther protocol. The results are provided in tab. 5.15 along with the description of the claims.

Claim 1: XORed message of location hash and timediff hash is maintained as a secret Current $location$ and δt_{diff} values from the configuration phase are xored together and sent to the destination base station as a request for key update. This request is message is signed by the private key of source base station to generate a value and encrypted by the public key of destination for confidentiality. This is evaluated and verified using Scyther and no attacks were found.

Claim 2: Confidentiality of hash function used Hash function is decided before

Table 5.15: Security Analysis of the SEA Framework using Scyther during Key Update Phase

Role	Sl.Num.	Claim	Status	Comment
β_i	1	secret $\{(location) \oplus sk_{\beta_i\beta_j}\}$	Ok	No Attacks
	2	Alive	Ok	No Attacks
	3	Weakagree	Ok	No Attacks
	4	Niagree	Ok	No Attacks
	5	Nisynch	Ok	No Attacks
β_j	1	secret $\{(location) \oplus sk_{\beta_i\beta_j}\}$	Ok	No Attacks
	2	Alive	Ok	No Attacks
	3	Weakagree	Ok	No Attacks
	4	Niagree	Ok	No Attacks
	5	Nisynch	Ok	No Attacks

dispersing the base stations into the network. So choice of a particular hash function is not decided in the communication, thus keeping it a secret.

Claim 3: Aliveness of both the base stations Both the neighboring base stations are believed to be alive in this configuration phase throughout the process. This claim is validated and verified using Scyther protocol analyzer tool. This protocol guarantees the aliveness of one base station to the other base station.

Claim 4: Weakagree The proposed protocol also guarantees the weakagreement between neighboring base stations. The source base station is running the proposed scheme with the neighboring base station, likewise the neighbouring base station is running the proposed scheme with the source base station. Communication is not affected by any adversary during the operation of the proposed scheme. This ensures and satisfies the weakagree claim. No adversary can initiate the start of the protocol by sending a key update request message.

Claim 5: Presence of Niagree between the base stations Niagree or non-injective agreement implies that during the activity of the proposed scheme, the source and the neighbor base station agree to any data exchange. The source base station can securely transmit information to the neighborbase station during the operation of the proposed scheme and vice-versa. If Niagree fails, then we can infer that a man-in-the-middle attack occurs during protocol execution. Our argument, however, is evaluated using Scyther protocol analyzer.

Claim 6: Ensuring Nisynch during the key update phase Nisynch or non-injective

synchronization is true if all steps before the claim are carried out in accordance with the proposed scheme definition. Nisynch ensures authentication process during the key update phase, there are no synchronization issues between source and neighbour base station. Nisynch further demands that the occurrence ofeach message isfollowed by the event that the message was sent, to ensure synchronization. Nisynch in the proposed scheme is validated using the Scyther protocol analyzer.

Commitments Validation

Commitments are used to achieve data integrity along with the use of logical clocks concept. Source base station makes sure that its replica data stored in neighboring base stations matches its data. This is accomplished by sending a commitment request to the destination base station containing the Ids of sensor nodes along with the encryption of location with source base stations private key to generate a sign. This request is satisfied by the destination base station by sending hash of sensor Ids along with respective logical clock values plus the hash of location signed with its private key. There is a chance of an attack using this approach, when the intruder has initial knowledge of the network as per given fig. 5.11.

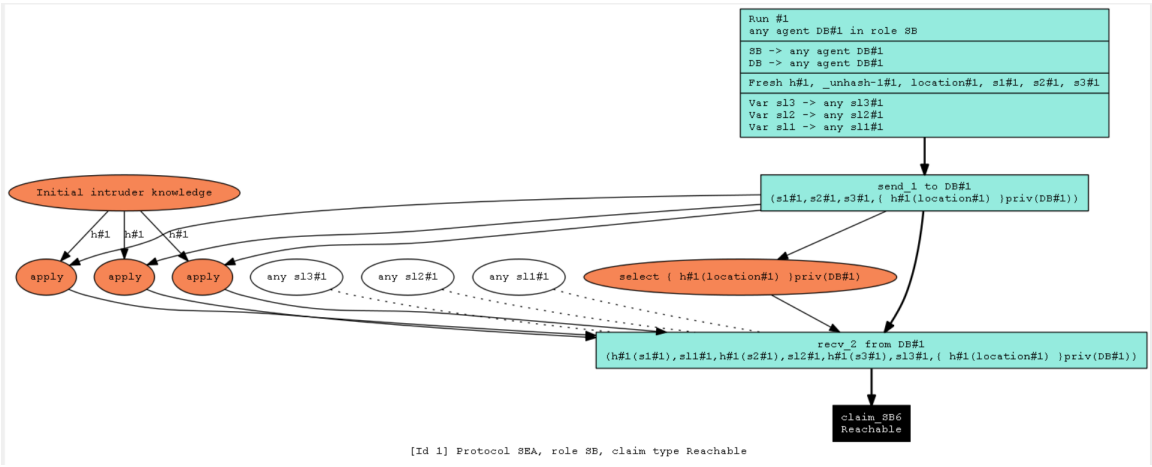

Figure 5.11: Scyther analysis displays possible security issue during information exchange - proposed commitments validation fails

Digital signature is used to verify source base station and is directly sent to the destination base station without any encryption of the data, which causes the following attack. As commitments are sent very frequently, In a dynamic sensor network,

Table 5.16: Security Analysis of the SEA Framework using Scyther during Key Commitments Validation with Public Key Encryption

Role	Sl.Num.	Claim	Status	Comment
β_i	1	secret *location*	Ok	No Attacks
	2	Alive	Ok	No Attacks
	3	Weakagree	Ok	No Attacks
	4	Niagree	No	at least 1 Attack
	5	Nisynch	No	at least 1 Attack
β_j	1	secret *location*	Ok	No Attacks
	2	Alive	Ok	No Attacks
	3	Weakagree	Ok	No Attacks
	4	Niagree	Ok	No Attacks
	5	Nisynch	Ok	No Attacks

these commitments do not require Public key encryption every time. Considering the present setup of the network, we are prioritizing efficiency over security for this section. However, semi dynamic or static network needs public key encryption as location and other parameters do not change. While also considering the system capacity as it takes considerable amount of time to encrypt and decrypt these messages. In case of more security requirement, the request message can be entirely encrypted with the public key of destination as shown in eq. 5.1, and eq. 5.2.

$$reqCommitment(\beta_i, \beta_j) = ENC_{pub_{\beta_j}}(s_1, s_2, ..., s_n, ENC_{priv_{\beta_i}}(h(location))) \quad (5.1)$$

$$resCommitment(\beta_j, \beta_i) = ENC_{pub_{\beta_i}}(h(s_1), lc(s_1), ..., h(s_n), lc(s_n), ENC_{priv_{\beta_j}}(h(location))) \quad (5.2)$$

The response message can be encrypted by the public key of source base station. This ensures security, although frequent use of such encrypted commitment request and response messages reduces the efficiency of the entire system which is not desirable. Tab. 5.16 shows different claims resulted by the use of this approach.

Claim 1: location is kept secret Hash of location is obtained by the destination base station from the configuration phase, this is used for authentication purpose in the commitments process. The source base station sends the hash of location signed by its private key for authentication. The response is sent by signing using private

key of neighbor base station. As hash of the location is being sent everytime, location is always a secret.

Claim 2: Aliveness of both parties Both the neighboring base stations are believed to be alive in this commitment phase throughout the process. This claim is validated and verified using Scyther protocol analyzer tool. This protocol guarantees the aliveness of one base station to the other base station.

Claim 3: Niagree No session establishment in here from source to destination, no public key encryption on outer package, it is assumed that network is dynamic and frequently changing so multiple publickey encryption operation affects the performance of the network. However from destination to source there is no problem with non injective agreement. The primary reason is usage of hashed message and digital signature.

Claim 4: Nisynch No synchronization from source to destination because it is assumed that commitments is a part of the entire communication. However, for experimental purposes a separate program on commitments is presented here, and therefore, the current scenario does not indicate any synchronization between the two parties.

Chapter 6

Conclusion

6.1 Conclusion

Wireless sensor networks are designed to collect physical information using tiny wireless sensors in different environments, including forest fires, battlefield monitoring and home sentry systems. Such technological innovations and advancements have brought forward new demands for data availability, immutability and network confidentiality.

This research introduces SEA, a reliable and secure framework for dynamic wireless sensor networks. SEA ensures data storage robustness by dynamically handling data duplication in base stations across the network. We suggest a replication strategy that satisfies both availability and performance requirements while taking memory requirements into account. It operates in an optimal way through efficient memory management under data loss conditions and reduces the overhead of distributed messages. The implemented scheme is used not only to replicate data, but to achieve data integrity in different neighbouring base stations using the logical clock principle. In addition, the proposed data replication scheme allows to reduce network traffic that has a significant impact on power consumption. The framework could be designed reconfigurable to accommodate changes such as replacing symmetric key encryption with asymmetric key encryption to enhance the security feature of the framework.

The SEA framework operates in a secure manner, which is achieved by a well designed configuration phase, key update phase, memory management phase and commitment phase. In addition, the proposed framework offers efficient key management, neighbour selection, key updating mechanism and the concept of logical clocks. This framework guarantees freshness of data and efficiency in data synchronisation among all base stations. In the past, insufficient memory and limited power did not allow low energy devices to execute resource intensive encryption techniques, however, due to emerging cryptographic technologies and the latest hardware innovation, this research work uses asymmetric key encryption.

In this research, the SEA framework is validated using computer emulation of dynamic wireless sensor network. SEA is considered efficient as the memory management between base stations is represented using computer simulation and the performance evaluation in terms of memory and communication overhead is compared with blockchain technology. The computer emulation exhibits that the proposed framework efficiently engages and disengages memory, which is beneficial for any dynamic wireless sensor applications such as smart hospitals, military applications etc. The outcome indicates promising results which helps in achieving efficiency of the network. Furthermore, formal security analysis performed by Scyther protocol analyzer indicates that the proposed framework can withstand numerous attacks. One key concept that the SEA framework highlights is, inclusion of variable replication factor, which ensures availability of data.

6.2 Future Work

The data replication of the WSN system in this scheme is set up in a static manner when the network is deployed. The replication factor remains unaltered until an individual manually interferes to adjust the number of replicas. As future work, it would be interesting to assess the data distribution efficiency considering dynamic replication factor which changes according to the network needs.

The designed SEA framework is validated by performing an efficiency comparison with blockchain technique. Operation of Radix DLT motivates the workflow of the current research, therefore the performance of SEA framework should be validated against Radix DLT. However, due to unavailability of the aforesaid technology, this validation can be considered as a future work. For the current scope of research we have considered blockchain for availability and efficiency comparison, however, to design a more robust framework we will evaluate the security of the current framework against blockchain and Radix DLT. Furthermore, a hardware implementation of SEA framework memory management could be considered as potential research area because of fast development of storage technologies.

Appendix I

Scenario 2

Scenario 2 Files are added in this section.

Table 1: Config CSV file entry based on scenario 2

T	NumBS	NumSN	BaseStationMemory	SensorMemory	ReplicationFactor
t0	5	15	256	8	4
t1	5	15	256	8	4
t2	5	15	256	8	4
t3	5	15	256	8	4
t4	5	15	256	8	4
t5	5	15	256	8	4
t6	5	15	256	8	4
t7	5	15	256	8	4
t8	5	15	256	8	4
t9	5	15	256	8	4
t10	5	15	256	8	4
t11	5	15	256	8	4
t12	5	15	256	8	4
t13	5	15	256	8	4
t14	5	15	256	8	4
t15	5	15	256	8	4

Table 2: Sample neighbour file data

t_i	ID_β	No. of Neighbours (k)	Neighbours (β_k)

```
t0, 1, 3, [3, 4, 2]
t0, 2, 3, [0, 3, 4]
t0, 4, 3, [2, 1, 0]
t0, 3, 3, [1, 0, 2]
t0, 0, 3, [1, 3, 4]
t1, 1, 3, [3, 4, 2]
t1, 2, 3, [0, 3, 4]
t1, 4, 3, [2, 1, 0]
t1, 3, 3, [1, 0, 2]
t1, 0, 3, [1, 3, 4]
t2, 1, 3, [3, 4, 2]
t2, 2, 3, [0, 3, 4]
t2, 4, 3, [2, 1, 0]
t2, 3, 3, [1, 0, 2]
t2, 0, 3, [1, 3, 4]
t3, 1, 3, [3, 4, 2]
t3, 2, 3, [0, 3, 4]
t3, 4, 3, [2, 1, 0]
t3, 3, 3, [1, 0, 2]
t3, 0, 3, [1, 3, 4]
t4, 1, 3, [3, 4, 2]
t4, 2, 3, [0, 3, 4]
t4, 4, 3, [2, 1, 0]
t4, 3, 3, [1, 0, 2]
t4, 0, 3, [1, 3, 4]
t5, 1, 3, [3, 4, 2]
t5, 2, 3, [0, 3, 4]
t5, 4, 3, [2, 1, 0]
t5, 3, 3, [1, 0, 2]
t5, 0, 3, [1, 3, 4]
t6, 1, 3, [3, 4, 2]
t6, 2, 3, [0, 3, 4]
t6, 4, 3, [2, 1, 0]
t6, 3, 3, [1, 0, 2]
t6, 0, 3, [1, 3, 4]
```

Table 3: Memory file entry based on scenario 2

T	BS ID	Occupied memory	Free Memory
t0	0	112	144
t1	0	224	32
t2	0	112	144
t3	0	224	32
t4	0	112	144
t5	0	224	32
t0	1	96	160
t1	1	192	64
t2	1	96	160
t3	1	192	64
t4	1	96	160
t5	1	192	64
t0	2	88	168
t1	2	176	80
t2	2	88	168
t3	2	176	80
t4	2	88	168
t5	2	176	80
t0	3	80	176
t1	3	160	96
t2	3	240	16
t3	3	80	176
t4	3	160	96
t5	3	240	16
t0	4	104	152
t1	4	208	48
t2	4	104	152
t3	4	208	48
t4	4	104	152
t5	4	208	48

Table 4: Sample sensor data for Scenario 2

T	0	1	2	3	4	5	6
t0	1010010	110	1111	101010	110111	10011110	11011110
t1	111	1110	1010101	10100100	10001000	11110001	11010110
t2	10010001	11111111	11011110	110011	10101010	11100	11111110
t3	10101011	10011000	10010100	11010101	1000001	1101011	11010010
t4	10001001	1100001	11001101	10011110	11011	100101	10001111
t5	1011110	111110	1110010	110011	10	1111010	10011000
t6	10101101	1111000	1111011	1101000	1101110	11000101	11100101
t7	111110	100110	1010100	100100	11100010	10010	11010
t8	10001010	10000001	10100	101110	10110000	10111000	1010101
t9	111	1001011	11011110	10010010	1000110	10111111	10101010
t10	10100110	10110101	11000	1111110	11101101	11011100	11010111
t11	10011010	10011011	10100101	10100111	10100	1110111	11000111
t12	1011101	10001110	10100101	1010001	11101010	1101	11110000
t13	11100000	1011100	10101111	1010010	11110011	10001111	11000010
t14	11100101	11011010	11000010	1011011	10000100	1100001	10010000
t15	10001111	10110111	1010010	111010	10001000	10111010	10000010
t16	1010100	1001100	10111011	1100111	11110111	1011001	1101111
t17	11010	10100010	1011110	110	1111110	11011110	10101001
t18	1111110	100011	111101	1010110	10010001	1011101	10000101
t19	11011010	1110100	11001001	11111110	10100010	11001101	11100110
t20	1110000	11010101	11101110	10111101	10100110	10100100	11010111

Table 5: Sample data of Topology.txt file

```
T, BS ID, Local_Sensors, Remote_Sensors
t0, 0, [6, 13, 1], [9, 4, 14, 5, 7, 10, 0, 2, 8, 3, 12]
t0, 1, [9, 4, 14, 5], [7, 10, 0, 2, 8, 3, 12, 11]
t0, 2, [11], [6, 13, 1, 7, 10, 0, 2, 8, 3, 12]
t0, 3, [7, 10], [9, 4, 14, 5, 6, 13, 1, 11]
t0, 4, [0, 2, 8, 3, 12], [11, 9, 4, 14, 5, 6, 13, 1]
t1, 0, [6, 13, 1], [9, 4, 14, 5, 7, 10, 0, 2, 8, 3, 12]
t1, 1, [9, 4, 14, 5], [7, 10, 0, 2, 8, 3, 12, 11]
t1, 2, [11], [6, 13, 1, 7, 10, 0, 2, 8, 3, 12]
t1, 3, [7, 10], [9, 4, 14, 5, 6, 13, 1, 11]
t1, 4, [0, 2, 8, 3, 12], [11, 9, 4, 14, 5, 6, 13, 1]
t2, 0, [6, 13, 1], [9, 4, 14, 5, 7, 10, 0, 2, 8, 3, 12]
t2, 1, [9, 4, 14, 5], [7, 10, 0, 2, 8, 3, 12, 11]
t2, 2, [11], [6, 13, 1, 7, 10, 0, 2, 8, 3, 12]
t2, 3, [7, 10], [9, 4, 14, 5, 6, 13, 1, 11]
t2, 4, [0, 2, 8, 3, 12], [11, 9, 4, 14, 5, 6, 13, 1]
t3, 0, [6, 13, 1], [9, 4, 14, 5, 7, 10, 0, 2, 8, 3, 12]
t3, 1, [9, 4, 14, 5], [7, 10, 0, 2, 8, 3, 12, 11]
t3, 2, [11], [6, 13, 1, 7, 10, 0, 2, 8, 3, 12]
t3, 3, [7, 10], [9, 4, 14, 5, 6, 13, 1, 11]
t3, 4, [0, 2, 8, 3, 12], [11, 9, 4, 14, 5, 6, 13, 1]
t4, 0, [6, 13, 1], [9, 4, 14, 5, 7, 10, 0, 2, 8, 3, 12]
t4, 1, [9, 4, 14, 5], [7, 10, 0, 2, 8, 3, 12, 11]
t4, 2, [11], [6, 13, 1, 7, 10, 0, 2, 8, 3, 12]
t4, 3, [7, 10], [9, 4, 14, 5, 6, 13, 1, 11]
t4, 4, [0, 2, 8, 3, 12], [11, 9, 4, 14, 5, 6, 13, 1]
t5, 0, [6, 13, 1], [9, 4, 14, 5, 7, 10, 0, 2, 8, 3, 12]
t5, 1, [9, 4, 14, 5], [7, 10, 0, 2, 8, 3, 12, 11]
t5, 2, [11], [6, 13, 1, 7, 10, 0, 2, 8, 3, 12]
t5, 3, [7, 10], [9, 4, 14, 5, 6, 13, 1, 11]
t5, 4, [0, 2, 8, 3, 12], [11, 9, 4, 14, 5, 6, 13, 1]
t6, 0, [6, 13, 1], [9, 4, 14, 5, 7, 10, 0, 2, 8, 3, 12]
t6, 1, [9, 4, 14, 5], [7, 10, 0, 2, 8, 3, 12, 11]
t6, 2, [11], [6, 13, 1, 7, 10, 0, 2, 8, 3, 12]
t6, 3, [7, 10], [9, 4, 14, 5, 6, 13, 1, 11]
t6, 4, [0, 2, 8, 3, 12], [11, 9, 4, 14, 5, 6, 13, 1]
```

Table 6: Sample log file

T	BS ID	
t2	0	complete data flushed
t2	1	complete data flushed
t2	2	complete data flushed
t2	4	complete data flushed
t3	3	complete data flushed
t4	0	complete data flushed
t4	1	complete data flushed
t4	2	complete data flushed
t4	4	complete data flushed
t6	0	complete data flushed
t6	1	complete data flushed
t6	2	complete data flushed
t6	3	complete data flushed
t6	4	complete data flushed
t8	0	complete data flushed
t8	1	complete data flushed
t8	2	complete data flushed
t8	4	complete data flushed
t9	3	complete data flushed
t10	0	complete data flushed
t10	1	complete data flushed
t10	2	complete data flushed
t10	4	complete data flushed

Scenario 3

Scenario 3 Files are added in this section.

Table 7: Config CSV file entry based on scenario 3

T	NumBS	NumSN	BaseStationMemory	SensorMemory	ReplicationFactor
t0	10	30	256	8	4
t1	10	30	256	8	4
t2	10	30	256	8	4
t3	10	30	256	8	4
t4	10	30	256	8	4
t5	10	30	256	8	4
t6	10	30	256	8	4
t7	10	30	256	8	4
t8	10	30	256	8	4
t9	10	30	256	8	4
t10	10	30	256	8	4
t11	10	30	256	8	4
t12	10	30	256	8	4
t13	10	30	256	8	4
t14	10	30	256	8	4
t15	10	30	256	8	4
t16	10	30	256	8	4
t17	10	30	256	8	4
t18	10	30	256	8	4
t19	10	30	256	8	4
t20	10	30	256	8	4

Table 8: Sample neighbour file data

t_i	ID_β	No. of Neighbours (k)	Neighbours (β_k)

```
t0, 7, 3, [8, 6, 3]
t0, 8, 3, [7, 5, 3]
t0, 9, 3, [1, 8, 4]
t0, 2, 3, [8, 6, 5]
t0, 3, 3, [0, 9, 6]
t0, 4, 3, [2, 1, 3]
t0, 5, 3, [7, 2, 0]
t0, 0, 3, [9, 4, 5]
t0, 1, 3, [2, 9, 0]
t0, 6, 3, [4, 1, 7]
t1, 7, 3, [8, 6, 3]
t1, 8, 3, [7, 5, 3]
t1, 9, 3, [1, 8, 4]
t1, 2, 3, [8, 6, 5]
t1, 3, 3, [0, 9, 6]
t1, 4, 3, [2, 1, 3]
t1, 5, 3, [7, 2, 0]
t1, 0, 3, [9, 4, 5]
t1, 1, 3, [2, 9, 0]
t1, 6, 3, [4, 1, 7]
t2, 7, 3, [8, 6, 3]
t2, 8, 3, [7, 5, 3]
t2, 9, 3, [1, 8, 4]
t2, 2, 3, [8, 6, 5]
t2, 3, 3, [0, 9, 6]
t2, 4, 3, [2, 1, 3]
t2, 5, 3, [7, 2, 0]
t2, 0, 3, [9, 4, 5]
t2, 1, 3, [2, 9, 0]
t2, 6, 3, [4, 1, 7]
t3, 7, 3, [8, 6, 3]
t3, 8, 3, [7, 5, 3]
t3, 9, 3, [1, 8, 4]
t3, 2, 3, [8, 6, 5]
t3, 3, 3, [0, 9, 6]
t3, 4, 3, [2, 1, 3]
t3, 5, 3, [7, 2, 0]
t3, 0, 3, [9, 4, 5]
t3, 1, 3, [2, 9, 0]
t3, 6, 3, [4, 1, 7]
```

Table 9: Memory file entry based on scenario 3

T	BS ID	Occupied memory	Free Memory
t0	0	96	160
t1	0	192	64
t2	0	96	160
t3	0	192	64
t0	1	96	160
t1	1	192	64
t2	1	96	160
t3	1	192	64
t0	2	120	136
t1	2	240	16
t2	2	120	136
t3	2	240	16
t0	3	88	168
t1	3	176	80
t2	3	88	168
t3	3	176	80
t0	4	64	192
t1	4	128	128
t2	4	192	64
t3	4	256	0
t0	5	112	144
t1	5	224	32
t2	5	112	144
t3	5	224	32
t0	6	80	176
t1	6	160	96
t2	6	240	16
t3	6	80	176
t0	7	104	152
t1	7	208	48
t2	7	104	152
t3	7	208	48
t0	8	104	152
t1	8	208	48
t2	8	104	152
t3	8	208	48
t0	9	96	160
t1	9	192	64
t2	9	96	160
t3	9	192	64

Table 10: Sample sensor data for Scenario 3

T	0	1	2	3	4	5	6
t0	1010010	110	1111	101010	110111	10011110	11011110
t1	10010001	11111111	11011110	110011	10101010	11100	11111110
t2	10001001	1100001	11001101	10011110	11011	100101	10001111
t3	10101101	1111000	1111011	1101000	1101110	11000101	11100101
t4	10001010	10000001	10100	101110	10110000	10111000	1010101
t5	10100110	10110101	11000	1111110	11101101	11011100	11010111
t6	1011101	10001110	10100101	1010001	11101010	1101	11110000
t7	11100101	11011010	11000010	1011011	10000100	1100001	10010000
t8	1010100	1001100	10111011	1100111	11110111	1011001	1101111
t9	1111110	100011	111101	1010110	10010001	1011101	10000101
t10	1110000	11010101	11101110	10111101	10100110	10100100	11010111
t11	10010	11000100	1110001	11011001	10100000	1110100	11010000
t12	111000	11000	1110110	1110010	1100011	1001111	100
t13	110011	1111010	10000011	111001	111100	10100000	11110100
t14	11111011	11101011	110110	110	1111110	11101000	10111000
t15	10001111	10101111	101011	1010001	1101	10111101	11010001
t16	1011101	1101010	111100	10011111	11011010	11100	11000111
t17	10111001	1000011	11111000	1010101	1100111	1000100	1001010
t18	11000	11111100	1000110	1001101	1010010	1101011	11111110
t19	10101010	10001010	11111001	110100	1111110	1010110	1111
t20	10011100	11111011	100110	1011101	11111100	11110010	10100010

Table 11: Sample data of Topology.txt file

```
T, BS ID, Local_Sensors, Remote_Sensors
t0, 0, [2, 17, 8], [7, 1, 27, 15, 9, 3, 24, 11, 4]
t0, 1, [12], [25, 18, 28, 20, 7, 1, 27, 15, 2, 17, 8]
t0, 2, [25, 18, 28, 20], [5, 6, 19, 13, 14, 22, 26, 21, 24, 11, 4]
t0, 3, [10], [2, 17, 8, 7, 1, 27, 15, 22, 26, 21]
t0, 4, [9, 3], [25, 18, 28, 20, 12, 10]
t0, 5, [24, 11, 4], [23, 16, 29, 0, 25, 18, 28, 20, 2, 17, 8]
t0, 6, [22, 26, 21], [9, 3, 12, 23, 16, 29, 0]
t0, 7, [23, 16, 29, 0], [5, 6, 19, 13, 14, 22, 26, 21, 10]
t0, 8, [5, 6, 19, 13, 14], [23, 16, 29, 0, 24, 11, 4, 10]
t0, 9, [7, 1, 27, 15], [12, 5, 6, 19, 13, 14, 9, 3]
t1, 0, [2, 17, 8], [7, 1, 27, 15, 9, 3, 24, 11, 4]
t1, 1, [12], [25, 18, 28, 20, 7, 1, 27, 15, 2, 17, 8]
t1, 2, [25, 18, 28, 20], [5, 6, 19, 13, 14, 22, 26, 21, 24, 11, 4]
t1, 3, [10], [2, 17, 8, 7, 1, 27, 15, 22, 26, 21]
t1, 4, [9, 3], [25, 18, 28, 20, 12, 10]
t1, 5, [24, 11, 4], [23, 16, 29, 0, 25, 18, 28, 20, 2, 17, 8]
t1, 6, [22, 26, 21], [9, 3, 12, 23, 16, 29, 0]
t1, 7, [23, 16, 29, 0], [5, 6, 19, 13, 14, 22, 26, 21, 10]
t1, 8, [5, 6, 19, 13, 14], [23, 16, 29, 0, 24, 11, 4, 10]
t1, 9, [7, 1, 27, 15], [12, 5, 6, 19, 13, 14, 9, 3]
t2, 0, [2, 17, 8], [7, 1, 27, 15, 9, 3, 24, 11, 4]
t2, 1, [12], [25, 18, 28, 20, 7, 1, 27, 15, 2, 17, 8]
t2, 2, [25, 18, 28, 20], [5, 6, 19, 13, 14, 22, 26, 21, 24, 11, 4]
t2, 3, [10], [2, 17, 8, 7, 1, 27, 15, 22, 26, 21]
t2, 4, [9, 3], [25, 18, 28, 20, 12, 10]
t2, 5, [24, 11, 4], [23, 16, 29, 0, 25, 18, 28, 20, 2, 17, 8]
t2, 6, [22, 26, 21], [9, 3, 12, 23, 16, 29, 0]
t2, 7, [23, 16, 29, 0], [5, 6, 19, 13, 14, 22, 26, 21, 10]
t2, 8, [5, 6, 19, 13, 14], [23, 16, 29, 0, 24, 11, 4, 10]
t2, 9, [7, 1, 27, 15], [12, 5, 6, 19, 13, 14, 9, 3]
t3, 0, [2, 17, 8], [7, 1, 27, 15, 9, 3, 24, 11, 4]
t3, 1, [12], [25, 18, 28, 20, 7, 1, 27, 15, 2, 17, 8]
t3, 2, [25, 18, 28, 20], [5, 6, 19, 13, 14, 22, 26, 21, 24, 11, 4]
t3, 3, [10], [2, 17, 8, 7, 1, 27, 15, 22, 26, 21]
t3, 4, [9, 3], [25, 18, 28, 20, 12, 10]
t3, 5, [24, 11, 4], [23, 16, 29, 0, 25, 18, 28, 20, 2, 17, 8]
t3, 6, [22, 26, 21], [9, 3, 12, 23, 16, 29, 0]
t3, 7, [23, 16, 29, 0], [5, 6, 19, 13, 14, 22, 26, 21, 10]
t3, 8, [5, 6, 19, 13, 14], [23, 16, 29, 0, 24, 11, 4, 10]
t3, 9, [7, 1, 27, 15], [12, 5, 6, 19, 13, 14, 9, 3]
```

Table 12: Sample log file

T	BS ID	
t2	0	complete data flushed
t2	1	complete data flushed
t2	2	complete data flushed
t2	3	complete data flushed
t2	5	complete data flushed
t2	7	complete data flushed
t2	8	complete data flushed
t2	9	complete data flushed
t3	6	complete data flushed
t4	0	complete data flushed
t4	1	complete data flushed
t4	2	complete data flushed
t4	3	complete data flushed
t4	4	complete data flushed
t4	5	complete data flushed
t4	7	complete data flushed
t4	8	complete data flushed
t4	9	complete data flushed
t6	0	complete data flushed
t6	1	complete data flushed
t6	2	complete data flushed
t6	3	complete data flushed
t6	5	complete data flushed
t6	6	complete data flushed
t6	7	complete data flushed
t6	8	complete data flushed
t6	9	complete data flushed

Bibliography

[1] A Very Brief History Of Blockchain Technology Everyone Should Read. https://www.forbes.com/sites/bernardmarr/2018/02/16/a-very-brief-history-of-blockchain-technology-everyone-should-read. [Online; accessed Feb-2020].

[2] Benefits of Elliptic Curve Cryptography. Technical report, 2012.

[3] Mohammad Reza Akhondi, Alex Talevski, Simon Carlsen, and Stig Petersen. Applications of wireless sensor networks in the oil, gas and resources industries. In *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, pages 941–948, 2010.

[4] Reem Alattas. Detecting black-hole attacks in WSNs using multiple base stations and check agents. In *FTC 2016 - Proceedings of Future Technologies Conference*, pages 1020–1024. Institute of Electrical and Electronics Engineers Inc., jan 2017.

[5] Osama Moh d. Alia. Dynamic relocation of mobile base station in wireless sensor networks using a cluster-based harmony search algorithm. *Information Sciences*, 385-386:76–95, apr 2017.

[6] Waleed Alsalih, Selim Akl, and Hossam Hassanein. Placement of multiple mobile base stations in wireless sensor networks. In *ISSPIT 2007 - 2007 IEEE International Symposium on Signal Processing and Information Technology*, pages 229–233, 2007.

[7] S Raj Anand, Rama Chaithanya Tanguturi, and Soundara Rajan. Blockchain Based Packet Delivery Mechanism for WSN. *International Journal of Recent Technology and Engineering*, (2):2277–3878, 2019.

[8] James Aspnes. LogicalClocks. https://www.cs.yale.edu/homes/aspnes/pinewiki/LogicalClocks.html. [Online; accessed Feb-2020].

[9] oko Banur, Branimir Jakšić, Miloš Banur, and Sran Jović. An analysis of energy efficiency in Wireless Sensor Networks (WSNs) applied in smart agriculture. *Computers and Electronics in Agriculture*, 156:500–507, jan 2019.

[10] Stefano Basagni, M Yousof Naderi, Chiara Petrioli, and Dora Spenza. WIRELESS SENSOR NETWORKS WITH ENERGY HARVESTING. Technical report.

[11] Wasana Boonsong and Widad Ismail. Wireless Monitoring of Household Electrical Power Meter Using Embedded RFID with Wireless Sensor Network Platform. *International Journal of Distributed Sensor Networks*, 10(6):876914, jun 2014.

[12] Florian Cäsar, Daniel P Hughes, Josh Primero, and Stephen J Thornton. Cerberus A Parallelized BFT Consensus Protocol for Radix. Technical report.

[13] Omer Cayirpunar, Bulent Tavli, Esra Kadioglu-Urtis, and Suleyman Uludag. Optimal Mobility Patterns of Multiple Base Stations for Wireless Sensor Network Lifetime Maximization. *IEEE Sensors Journal*, 17(21):7177–7188, nov 2017.

[14] Sravanthi Chalasani and James M. Conrad. A survey of energy harvesting sources for embedded systems. In *Conference Proceedings - IEEE SOUTHEASTCON*, pages 442–447, 2008.

[15] Rishav Chatterjee and Rajdeep Chatterjee. An Overview of the Emerging Technology: Blockchain. In *Proceedings - 2017 International Conference on Computational Intelligence and Networks, CINE 2017*, pages 126–127. Institute of Electrical and Electronics Engineers Inc., mar 2018.

[16] Ruirui Chen, Yanjing Sun, Yan Chen, Xiaoguang Zhang, Song Li, and Zhi Sun. Energy Efficiency Analysis of Bidirectional Wireless Information and Power Transfer for Cooperative Sensor Networks. *IEEE Access*, 7:4905–4912, 2019.

[17] Yuequan Chen, Edward Chan, and Song Han. Energy Efficient Multipath Routing in Large Scale Sensor Networks with Multiple Sink Nodes. pages 390–399. 2005.

[18] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and Smart Contracts for the Internet of Things, 2016.

[19] Hilmi Egemen Ciritoglu, Leandro Almeida, Eduardo Almeida, Teodora Buda, John Murphy, and Christina Thorpe. Investigation of replication factor for performance enhancement in the hadoop distributed file system. pages 135–140, 04 2018.

[20] Marco Conoscenti, Antonio Vetro, and Juan Carlos De Martin. Blockchain for the Internet of Things: A systematic literature review. In *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*. IEEE Computer Society, jul 2017.

[21] Cas J. F. Cremers. The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols. In Aarti Gupta and Sharad Malik, editors, *Computer Aided Verification*, pages 414–418, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[22] Debasree Das, Zeenat Rehena, Sarbani Roy, and Nandini Mukherjee. Multiple-sink placement strategies in wireless sensor networks. In *2013 5th International Conference on Communication Systems and Networks, COMSNETS 2013*, 2013.

[23] S. Dey, Q. Ye, and S. Sampalli. Amlt: A mutual authentication scheme for mobile cloud computing. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom)*

and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pages 700–705, July 2018.

[24] Whitfield Diffie and Martin E Hellman. New Directions in Cryptography Invited Paper. Technical report.

[25] Ali Dorri, Salil S. Kanhere, and Raja Jurdak. Blockchain in internet of things: Challenges and Solutions. aug 2016.

[26] Ali Dorri, Salil S. Kanhere, Raja Jurdak, and Praveen Gauravaram. Blockchain for IoT security and privacy: The case study of a smart home. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2017*, pages 618–623. Institute of Electrical and Electronics Engineers Inc., may 2017.

[27] M.P. Durisic, Zhilbert Tafa, Goran Dimic, and Veljko Milutinovic. A survey of military applications of wireless sensor networks. pages 196–199, 01 2012.

[28] P. L. Fuhr. Wireless sensor networks for the monitoring and control of nuclear power plants. In *Industrial Wireless Sensor Networks: Monitoring, Control and Automation*, pages 125–154. Elsevier Inc., jan 2016.

[29] Xiong Gan, Hong Lu, and Guangyou Yang. Improving energy efficiency by optimizing relay nodes deployment in wireless sensor networks. In *2017 9th IEEE International Conference on Communication Software and Networks, ICCSN 2017*, volume 2017-Janua, pages 306–310. Institute of Electrical and Electronics Engineers Inc., dec 2017.

[30] Shashidhar Rao Gandham, Milind Dawande, Ravi Prakash, and S Venkatesan. Energy Efficient Schemes for Wireless Sensor Networks with Multiple Mobile Base Stations. Technical report.

[31] Shashidhar Rao Gandham, Milind Dawande, Ravi Prakash, and S. Venkatesan. Energy Efficient Schemes for Wireless Sensor Networks with Multiple Mobile Base Stations. In *GLOBECOM - IEEE Global Telecommunications Conference*, volume 1, pages 377–381, 2003.

[32] Kaushik Ghosh, Sarmistha Neogy, Pradip K. Das, and Mahima Mehta. Intrusion Detection at International Borders and Large Military Barracks with Multi-sink Wireless Sensor Networks: An Energy Efficient Solution. *Wireless Personal Communications*, 98(1):1083–1101, jan 2018.

[33] Vehbi C. Gungor and Gerhard P. Hancke. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *IEEE Transactions on Industrial Electronics*, 56(10):4258–4265, 2009.

[34] Guangjie Han, Mengting Xu, Yu He, Jinfang Jiang, James Adu Ansere, and W. Zhang. A dynamic ring-based routing scheme for source location privacy in wireless sensor networks. *Information Sciences*, 504:308–323, dec 2019.

[35] Thaier Hayajneh, Razvi Doomun, Ghada Al-Mashaqbeh, and Bassam J. Mohd. An energy-efficient and security aware route selection protocol for wireless sensor networks. *Security and Communication Networks*, 7(11):2015–2038, nov 2014.

[36] Dan Hughes. Tempo - Radix Knowledge Base. `https://docs.radixdlt.com/kb/learn/whitepapers/tempo`. [Online; accessed Feb-2020].

[37] Seyoung Huh, Sangrae Cho, and Soohyung Kim. Managing IoT devices using blockchain platform. In *International Conference on Advanced Communication Technology, ICACT*, pages 464–467. Institute of Electrical and Electronics Engineers Inc., mar 2017.

[38] Simona Ibba, Andrea Pinna, Matteo Seu, and Filippo Eros Pani. CitySense: Blockchain-oriented Smart Cities. In *ACM International Conference Proceeding Series*, volume Part F1299. Association for Computing Machinery, may 2017.

[39] Nazmul Islam. TOWARDS A SECURE AND ENERGY EFFICIENT WIRELESS SENSOR NETWORK USING BLOCKCHAIN AND A NOVEL CLUSTERING APPROACH. Technical report, 2018.

[40] S. R. Jino Ramson and D. Jackuline Moni. Applications of Wireless Sensor Networks - A survey. In *Proceedings of IEEE International Conference on Innovations in Electrical, Electronics, Instrumentation and Media Technology, ICIEEIMT 2017*, volume 2017-Janua, pages 325–329. Institute of Electrical and Electronics Engineers Inc., nov 2017.

[41] Don Johnson, Alfred Menezes, £, and Scott Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). Technical report.

[42] Ashish Joshi and Amar Kumar Mohapatra. Authentication protocols for wireless body area network with key management approach. *Journal of Discrete Mathematical Sciences and Cryptography*, 22(2):219–240, feb 2019.

[43] Vaidehi Joshi. Logical Time and Lamport Clocks (Part 2) - baseds - Medium. `https://medium.com/baseds/logical-time-and-lamport-clocks-part-2-272c097dcdda`. [Online; accessed Feb-2020].

[44] Ibrahim Kamel and Hussam Juma. A lightweight data integrity scheme for sensor networks. *Sensors*, 11(4):4118–4136, apr 2011.

[45] B. Kitchenham, B. Kitchenham, and S Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. 2007.

[46] Neal Koblitz and Alfred J Menezes. A Survey of Public-Key Cryptosystems. Technical report, 2004.

[47] Nallapaneni Manoj Kumar and Pradeep Kumar Mallick. Blockchain technology for security issues and challenges in IoT. In *Procedia Computer Science*, volume 132, pages 1815–1823. Elsevier B.V., 2018.

[48] Sergii Kushch and Francisco Prieto-Castrillo. A Rolling Blockchain for a Dynamic WSNs in a Smart City. jun 2018.

[49] Leslie Lamport. Operating R. Stockton Gaines Systems Editor Time, Clocks, and the Ordering of Events in a Distributed System. Technical report, 1978.

[50] Kristin Lauter. The Advantages of Elliptic Curve Cryptography for Wireless Security, feb 2004.

[51] Wen Hwa Liao and Chun Chu Chen. A multi-dimensional data storage algorithm in wireless sensor networks. In *Proceedings of the 2010 IEEE/IFIP Network Operations and Management Symposium, NOMS 2010*, pages 854–857. IEEE Computer Society, 2010.

[52] Yujin Lim, Hak-Man Kim, and Sanggil Kang. A Design of Wireless Sensor Networks for a Power Quality Monitoring System. *Sensors*, 10(11):9712–9725, nov 2010.

[53] Zhi Ting Lin, Jie Zheng, Yu Sheng Ji, Bao Hua Zhao, Yu Gui Qu, Xu Dong Huang, and Xiu Fang Jiang. EMMNet: Sensor networking for electricity meter monitoring. *Sensors*, 10(7):6307–6323, jul 2010.

[54] Julio López, Julio López, and Ricardo Dahab. An Overview of Elliptic Curve Cryptography. 22, 2000.

[55] J. Louw, G. Niezen, T. D. Ramotsoela, and A. M. Abu-Mahfouz. A key distribution scheme using elliptic curve cryptography in wireless sensor networks. In *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, pages 1166–1170, July 2016.

[56] Jun Luo, Jacques Panchard, Michal Piórkowski, Matthias Grossglauser, and Jean Pierre Hubaux. MobiRoute: Routing towards a mobile sink for improving lifetime in sensor networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4026 LNCS, pages 480–497. Springer Verlag, 2006.

[57] Lunda Machaya and Simon Tembo. A Study on Memory Management in Wireless Sensor Nodes during Key Agreement Generation. 6(2):19–23, 2016.

[58] Mohammad Matin and Md Islam. *Overview of Wireless Sensor Network*, pages 1–22. 09 2012.

[59] C. Meesookho, S. Narayanan, and C. S. Raghavendra. Collaborative classification applications in sensor networks. In *Proceedings of the IEEE Sensor Array and Multichannel Signal Processing Workshop*, volume 2002-Janua, pages 370–374. IEEE Computer Society, 2002.

[60] Kais Mekki, William Derigent, Eric Rondeau, and André Thomas. In-network data storage protocols for wireless sensor networks: A state-of-the-art survey. *International Journal of Distributed Sensor Networks*, 15(4):155014771983248, apr 2019.

[61] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 293 LNCS, pages 369–378. Springer Verlag, 1988.

[62] Evgeny Milanov. The RSA Algorithm. Technical report, 2009.

[63] David Mills, Kathy Wang, Brendan Malone, Anjana Ravi, Jeff Marquardt, Clinton Chen, Anton Badev, Timothy Brezinski, Linda Fahy, Kimberley Liao, Vanessa Kargenian, Max Ellithorpe, Wendy Ng, and Maria Baird. Distributed Ledger Technology in Payments, Clearing, and Settlement. *Finance and Economics Discussion Series*, 2016(095), dec 2016.

[64] Axel Moinet, Benoît Darties, and Jean-Luc Baril. Blockchain based trust & authentication for decentralized sensor networks. jun 2017.

[65] G Padmavathi and Mrs D Shanmugapriya. A Survey of Attacks, Security Mechanisms and Challenges in Wireless Sensor Networks. Technical Report 1, 2009.

[66] Manjiri Pathak. An Approach to Memory management in Wireless Sensor Networks. Technical report.

[67] Wint Yi Poe and Jens B Schmitt. Minimizing the Maximum Delay in Wireless Sensor Networks by Intelligent Sink Placement. Technical report.

[68] Boselin Prabhu and N. Balakumar. Highly Distributed and Energy Efficient Clustering Algorithm for Wireless Sensor Networks, nov 2016.

[69] Radix. Radix Platform - Radix Knowledge Base. `https://docs.radixdlt.com/kb/learn/platform`. [Online; accessed Feb-2020].

[70] Bushra Rashid and Mubashir Husain Rehmani. Applications of wireless sensor networks for urban areas: A survey, jan 2016.

[71] N. Gouthamsekhar Reddy, Neeranjan Chitare, and Srinivas Sampalli. Deployment of multiple base-stations in clustering protocols of wireless sensor networks (WSNs). In *Proceedings of the 2013 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2013*, pages 1003–1006, 2013.

[72] Ju Ren, Yaoxue Zhang, Ning Zhang, Deyu Zhang, and Xuemin Shen. Dynamic Channel Access to Improve Energy Efficiency in Cognitive Radio Sensor Networks. *IEEE Transactions on Wireless Communications*, 15(5):3143–3156, may 2016.

[73] Yongjun Ren, Yepeng Liu, Sai Ji, Arun Kumar Sangaiah, and Jin Wang. Incentive Mechanism of Data Storage Based on Blockchain for Wireless Sensor Networks. *Mobile Information Systems*, 2018, 2018.

[74] Kay Römer and Friedemann Mattern. The design space of wireless sensor networks, dec 2004.

[75] Wei She, Qi Liu, Zhao Tian, Jian Sen Chen, Bo Wang, and Wei Liu. Blockchain trust model for malicious node detection in wireless sensor networks. *IEEE Access*, 7:38947–38956, 2019.

[76] Mazn Adnan Shkoor. Everything You Need to Know About Public, Private, and Consortium Blockchain.

[77] Melanie Swan. Blockchain - Blueprint for a new economy - EPDF.PUB.

[78] Tim Swanson. Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems. Technical report, 2015.

[79] Bo Tang, Jin Wang, Xuehua Geng, Yuhui Zheng, and Jeong-Uk Kim. A Novel Data Retrieving Mechanism in Wireless Sensor Networks with Path-Limited Mobile Sink. Technical Report 3, 2012.

[80] Raja Jalees ul Hussen Khan, Zainib Noshad, Atia Javaid, Maheen Zahid, Ishtiaq Ali, and Nadeem Javaid. Node Recovery in Wireless Sensor Networks via Blockchain. In *Lecture Notes in Networks and Systems*, volume 96, pages 94–105. Springer, 2020.

[81] S. Viswanathan and A. Kannan. Elliptic key cryptography with Beta Gamma functions for secure routing in wireless sensor networks. *Wireless Networks*, 25(8):4903–4914, nov 2019.

[82] John Paul Walters, Zhengqiang Liang, Weisong Shi, and Vipin Chaudhary. Security in Distributed, Grid, and Pervasive Computing Yang Xiao. Technical report.

[83] Arvinderpal S. Wandert, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Proceedings - Third IEEE International Conference on Pervasive Computing and Communications, PerCom 2005*, volume 2005, pages 324–328, 2005.

[84] Haiyong Wang, Geng Yang, Jian Xu, Zhengyu Chen, Lei Chen, and Zhen Yang. A noval data collection approach for Wirelsee Sensor Networks. In *2011 International Conference on Electrical and Control Engineering, ICECE 2011 - Proceedings*, pages 4287–4290, 2011.

[85] Huaimin Wang, Zibin Zheng, Shaoan Xie, Hong-Ning Dai, and Xiangping Chen. Blockchain challenges and opportunities: a survey. *International Journal of Web and Grid Services*, 14:352 – 375, 10 2018.

[86] Yang Wang, Xiong Fu, and Liusheng Huang. A storage node placement algorithm in wireless sensor networks. In *4th International Conference on Frontier of Computer Science and Technology, FCST 2009*, pages 267–271, 2009.

[87] Jeremy Wohlwend. ELLIPTIC CURVE CRYPTOGRAPHY: PRE AND POST QUANTUM. Technical report.

[88] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. Blockchain Technology Overview.

[89] Jidian Yang, Shiwen He, Yang Xu, Linweiya Chen, and Ju Ren. A Trusted Routing Scheme Using Blockchain and Reinforcement Learning for Wireless Sensor Networks. *Sensors*, 19(4):970, feb 2019.

[90] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. 2008.

[91] Byrav ; Yong; Ramamurthy and Yuyan Xue. A Key Management Protocol for Wireless Sensor Networks with Multiple Base Stations. page 111, 2008.