ProBLeSS: A Proactive Blockchain based Spectrum Sharing Protocol against SSDF Attacks in Cognitive Radio IoBT Network

A Project Report

submitted by

G PRABHU (CS18M004)

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

JUNE 2020

Certificate

This is to certify that the thesis titled **ProBLeSS: A Proactive Blockchain based Spectrum Sharing Protocol against SSDF Attacks in Cognitive Radio IoBT Network**, submitted by **G Prabhu**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bonafide record of the research work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Chester Rebeiro
Project Guide
Professor
Dept. of Computer Science and Engineering
IIT Madras, 600 036

Place: Chennai

Date: June 15, 2020

DECLARATIONS

I declare that this project work titled **"ProBLeSS: A Proactive Blockchain based Spectrum Sharing Protocol against SSDF Attacks in Cognitive Radio IoBT Network"** submitted in partial fulfillment for the award of the degree of **Master of Technology** is a record of original research work carried out by me under the supervision of **Dr. Chester Rebeiro**, and has not formed the basis for the award of any degree, diploma, associate ship, fellowship, or other titles in this or any other Institution or University of higher learning. In keeping with the ethical practice in reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

G Prabhu
M. Tech
Dept. of Computer Science and Engineering
IIT Madras, 600 036

Place: Chennai

Date: June 15, 2020

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisor Dr. Chester Rebeiro for his continuous support for my MTech Project and related work, for his patience, motivation, and immense knowledge. His guidance has helped me in understanding the basics of research work has helped me in completion of the project and also in writing this report. I could not ask for a better advisor and mentor for my project.

Besides my advisor, I would also like to thank Col. Milan Patnaik for his idea and motivation for the project. He had been a pillar of support through out the project work.

I would also extend my sincere gratitude and thanks to Mr. Hari, Mr. Gokul Krishnan G and Mr. Tulasi Dwarakanath V, officials of RTS & IoT Group, CDAC, Bengaluru, who have advised me on the Ubimote-SEZ device and also provided device support throughout the project.

I would like to thank all my MTech classmates and RISE lab group for their constant support and motivation. I would cherish the memories associated with IIT Madras during my two years in this great institute.

Jai Hind

G Prabhu

IITM, Chennai.

ABSTRACT

KEYWORDS: Cognitive Radio (CR); Spectrum Sensing Data Falsification (SSDF); Denial of Service (DoS); Internet of Battlefield Things (IoBT); Blockchain;

Spectrum Sharing Data Falsification (SSDF) attacks can cause heavy performance degradation to Cognitive Radio (CR) based Internet of Battle Things (IoBT) networks. The challenge in such networks is to handle this security problem in real-time, in addition to the other important spectrum related tasks. This requires a robust CR architecture and protocol that can provide integrity and reliability of the spectrum sensing data being shared between Secondary Users (SUs) for collaborative spectrum decisions. We propose one such protocol called Proactive Blockchain based Spectrum Sharing Protocol (ProBLeSS) which leverages a blockchain to provide security against SSDF attacks in CR Networks (CRN). This protocol leverages the proven security features of Blockchain technology to handle and share spectrum data among the SUs and also prevent the network from SSDF attacks by malicious users. ProBLeSS algorithm was tested on a simulated network with an intelligent SSDF attack based on PROLEMus that has resulted in average 1.58 % increase in Channel Utilization, 7.59 % and 13.54 % reduction in Sensing Delay and Backoff Rate respectively when compared to PROLEMus under similar SSDF attack.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

RTFM Read the Fine Manual

NOTATION

α	Channel Utilization
β	Backoff Rte
γ	Sensing Delay

CHAPTER 1

INTRODUCTION

Future battlefield is envisaged to be a technologically advanced environment with complex command and control structure, where the entire hard power of a country can be applied at once and conflicts can be resolved in the shortest time frame. This implies that multiple fighting forces will be deployed in a small conflict zone and will have less time for detailed coordination. The task forces are one such entity that are to be prepared for any task at any location in a battlefield. The role and organization of these forces are dynamic in nature, accordingly they are trained and equipped. This kind of dynamic force need a network of devices that are smart and self adjusting to the varied organizational structure.

1.1 Cognitive Radio in Battlefield

The core elements of such a network will be the Internet of Battlefield Things (**IoBT**) consisting of armor, radios, weapons, and other objects that will connect soldiers with smart technology to provide extrasensory perception, increase situational awareness, endow fighters with prediction powers, provide better risk assessment, and develop shared intuitions. These IoBTs will be connected through a radio network. The use of radio in a battlefield environment involves practical challenges of frequency management due to spectrum scarcity. A promising solution to address the spectrum scarcity problem, that arises due to increase in such smart endpoints in a battlefield, is the use of Cognitive Radio (**CR**)[1] technology in the IoBT, leading to CR-based IoBT (CR-IoBT). The CR devices are intelligent that can adjust their transmission parameters according to the environment, thereby avoiding any last minute frequency allotment and coordination requirements. They can be deployed without any licensed frequency allotted to them in advance.

1.2 Spectrum Sensing in Cognitive Radios

Cognitive Radios communicate using the vacant frequency spots (Spectrum Holes) in the licensed EM spectrum. These vacant frequencies must be used without causing any interference to the Primary (Licensed) Users (PUs). Sensing the EM spectrum to identify these **Spectrum Holes** is a critical function of any CR and has considerable effect on the performance of a CR. One method to increase the performance of CR-IoBT is *Collaborative Spectrum Sensing* where a number of BEs sense the environment and send their observations to all other BEs or a Master BE [3]. BEs or the Master BE then mix the provided information to take the final decision regarding the presence or absence of PU transmissions and marks a channel as busy or idle. Decision cycle of a Cognitive Radio is given in Fig. 1.1. If a Master BE processes the information from all other BEs and makes a decision, then the spectrum sensing approach is called as Centralised Collaborative Spectrum Sensing. Otherwise, the spectrum sharing is called Distributed Collaborative Spectrum Sensing.

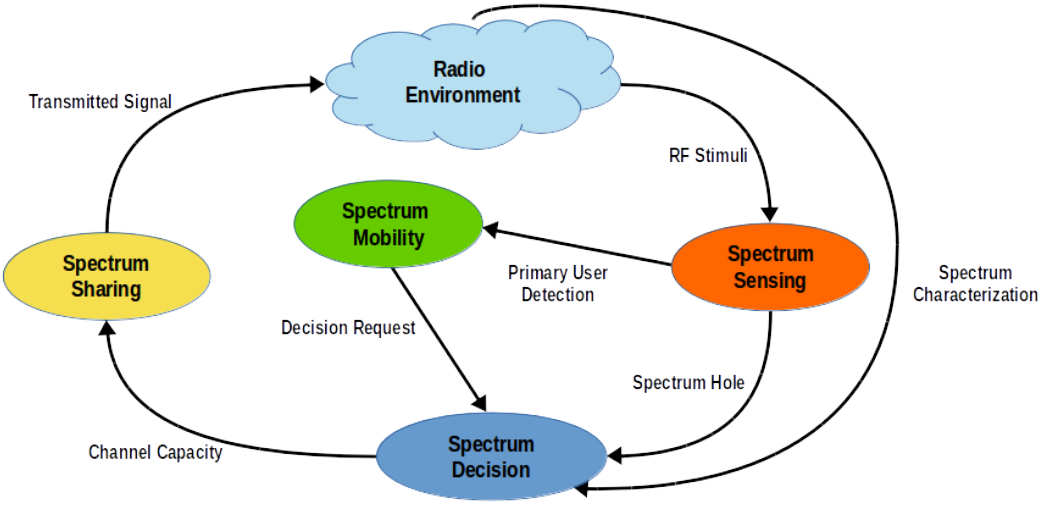

Figure 1.1: Cognitive Radio Cycle

1.3 SSDF Attack in Cognitive Radio

The CR device in an IoBT architecture will sense the network and detect free channels in the spatial and/or temporal domain and decide which free channel to use. Thus, spectrum sensing forms a crucial step in the cognitive cycle, wherein a BE needs to back-off when it senses a signal from PU. This opens up a large loop-hole for malicious

users who can act like BEs in an IoBT network and force BEs to vacate a specific band or can provide false observations hampering the spectrum decision making process. In addition, performance of any radio network is highly affected by typical problems in radio communication like fading, etc. Fading is the problem faced by the wireless communication channels where the expected gain of the channel is not able to be attained due to weather, screening by other objects, geographical position, etc. In Collaborative Spectrum Sensing scenario, any malicious BE can generate and share false observations to the other BEs thereby intentionally causing a Denial of Service(DoS). This degrades the reliability of collaborative spectrum sensing. This type of attack is called the **Spectrum Sensing Data Falsification (SSDF)** attack [3]. There are various other possible security threats to the CRN, however, we will address this particular attack in our work as it can cause a Denial of Service resulting in failure of a mission.

1.4 Blockchain against SSDF Attack

The spectrum data sensed and stored by the CRs has two major threats which will lead to the SSDF attacks. The threats are as follows.

1. **Integrity of the spectrum data**. Integrity of data stored in digital form can be achieved by maintaining a hash/digest of the data and verifying it while using the data. Blockchain is a chain of such data digests which makes it difficult for the attacker to change the data at any point in the chain. Hence the Spectrum Data of the CRs can be stored in Blockchain to ensure the integrity of the data.

2. **False spectrum data shared by Malicious CRs.** The Spectrum data shared among the CRs may be falsified if a node is compromised and it will affect the decision making of all the CRs. Hence the data that is shared by each CR will have to be validated before being committed into the block. This consensus among the CRs before committing the block and blacklisting a CR that is sending the false data is part of a blockchain implementation.

Blockchain technology provides a perfect solution to both the possible ways of SSDF attacks in CRs. In this project we evaluate the use Blockchain as a solution to the SSDF attacks in Cognitive Radios.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Cognitive Radio

Wireless communications has seen a rapid growth in the past two decades; the user base has seen an exponential growth. With a new technology lined up for implementation at every passing year, this has definitely changed the way in which people communicate. One important resource in wireless communications technology is the *Electro Magnetic (EM)* Spectrum. It is also considered as an expensive natural resource which is heavily regulated by governments. Though the EM spectrum/frequency seems to be unlimited, they are limited in the way we utilize them for various services. With growing number of applications and fixed amount of resources (EM Spectrum), the industry faced a resource crunch called *Spectrum Scarcity*. The allocation of frequency to the licensed users is still based on an auction. The licensed users holds the exclusive rights on this allocated spectrum. Though heavily regulated, the frequency spectrum is still being under utilized, as per the Federal Communications Commission (FCC), many portions of the allocated frequency spectrum is not utilized for a significant period of time i.e, temporal and geographical variations in utilization of assigned spectrum range from 15 to 85%.

The limited availability of EM spectrum and the difficulty in managing the spectrum usage has led to a new communication paradigm to exploit the existing EM spectrum dynamically. *Dynamic Spectrum Access (DSA)* is an efficient technique that can solve the spectrum scarcity problem. Here unlicensed users can opportunistically use the licensed spectrum bands without any interference to the licensed users. This could be achieved by using the *Software Defined Radios(SDR)* which can operate on any frequency band in a wide range of spectrum with minimum time delay. Cognitive Radio is one such SDR that can find the non-utilized frequencies in the spectrum and use them to communicate, also ensuring that the licensed users of that frequency is not affected. CRNs provide a high bandwidth and speed to wireless users using heterogeneous wireless architectures and DSA techniques.

2.2 Concepts and Terminologies of Cognitive Radio

To understand the working of Cognitive Radios, certain common terminologies has to be understood in detail. This section will cover some of the commonly used terminologies in CRs.

2.2.1 Primary Users

Primary Users (PU) are the users who holds the government allocated license to use a particular spectrum in a particular geographical area. These users always have the priority to use the spectrum at any point of time and interference in their usage of channels is against the law.

2.2.2 Secondary Users

Secondary Users (SU) are the Cognitive Radios who opportunistically use the unused spectrum for communication. This is achieved by constantly sensing the spectrum and predicting the availability of a frequency for future use. The usage of unused spectrum by the SU should not interfere with the operation of a PU. Hence the SUs must vacate the spectrum immediately on learning that a PU is trying to access the channel.

2.2.3 Spectrum Holes

A spectrum hole is defined as a frequency or a band of frequency that are not utilized by a PU at a certain time, or location, or polarization or code or the combination of these. Cognitive Radios operate mainly by locating these holes and communicating through these holes. Spectrum Sensing enables the CRs to identify a channel to be free or busy.

2.3 Tasks and Functions

Cognitive operation of a Cognitive radio requires extraordinary capabilities and specific functions that enable it to effectively use the spectrum. In this section we will see

some capabilities and functions that are required to be performed by the CRs.

2.3.1 Cognitive Capability

Cognitive capability refers to the ability of the radio device to achieve "awareness" of the surrounding environment by extracting useful knowledge about the radio environment in a smart and efficient manner, while exerting minimum interference to the surrounding users. This knowledge should comprise many dimensions, not only, frequency and time, but also space, power, interference levels, different codes, etc.

2.3.2 Reconfigurability

Reconfigurability enables the cognitive radio to use the gained knowledge by its cognitive capability to dynamically reconfigure its transmission/receive parameters to adapt with the radio environment variations. The embedded SDR in a cognitive radio makes this task easier by enabling the cognitive radio to dynamically configure a variety of parameters such as frequency of operation, modulation and coding scheme, etc. For reconfigurability, the software-defined radio entity performs this task. For other tasks of a cognitive kind, cognitive radio assigns them to signal-processing and machine-learning procedures.

2.3.3 Spectrum Sensing

First task of a Cognitive Radio is to detect/sense the frequency spectrum in its domain. Further operations will be based on this sensed data about the surrounding spectrum. CRs use various techniques to sense the EM spectrum and identify Spectrum Holes. There are various effective methods available to detect Spectrum Holes by a CR. A CR can individually detect the spectrum holes or it can cooperate with other CRs. In cooperative Spectrum sensing, the CRs share their sensed data with other CRs. This Spectrum information sharing can be Centralized (with a common CR) or Distributed (Each CR shares the data will all other devices). Classification of CRs based on Spectrum Sensing is given in Fig. 2.1

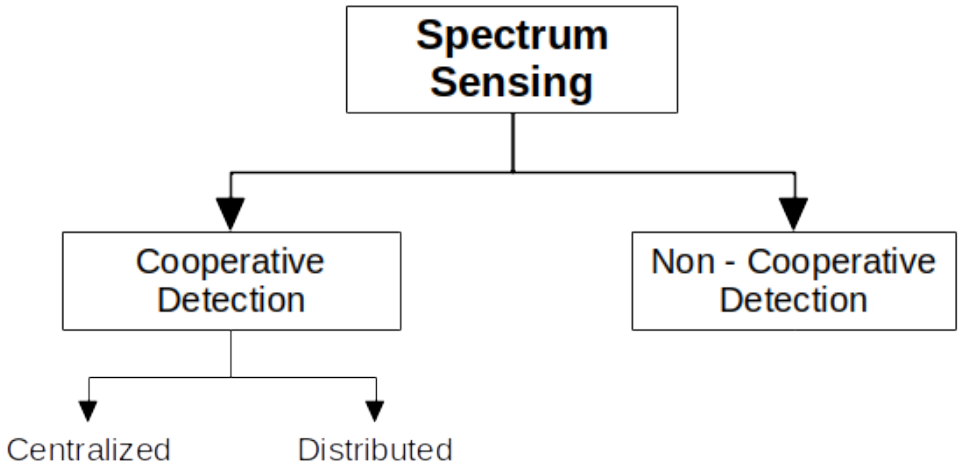

Figure 2.1: Spectrum Sensing Methods

2.3.4 Spectrum Analysis

The Cognitive part of the CRs is when they learn from the collected Spectrum Hole data to take decision on the likely channel that can be used communication by the CRs. The spectrum data is classified to enable the CRs to take decision on them. Classification will be on various parameters which will not be discussed in this work.

2.3.5 Spectrum Decision

Each CR will require different type of service from a particular spectrum like video, data, voice, etc. The parameters used by Spectrum Analysis will be helpful in enabling the CRs to take decision on a particular Spectrum Hole for its requirement. Spectrum Decision algorithm enables the CR to find a best fit spectrum for its service requirement.

2.3.6 Spectrum Mobility

The ability of a Cognitive Radio to detect and vacate the channel when the Primary User tries to access it is called Spectrum Mobility. Its considered most important capability, as the PU must not be interfered in its operation by the SUs.

2.4 Security Threats to Cognitive Radio

Cognitive Radios will able to communicate with other devices and provide the necessary Quality of Service (QoS) if there exists a mutual trust among the peers in the CRN. CRNs are subject to various security threats at different layers. We will some of the important security threats in this section.

2.4.1 Spectrum Sensing Data Falsification Attack

This attack is the transmission of false spectrum sensing data by malicious secondary users. SSDF are referred to such attacks where an attacker may send false local spectrum sensing results to a data collector, causing the data collector to make a wrong spectrum sensing decision. This attack is also known as the Byzantine Attack. It takes place when an attacker sends false local spectrum sensing data to its neighbours or to the fusion center. In a centralized CRN, a fusion center collects all the sensed data and then uses them to take a decision on which frequency bands are occupied and which are free. Cheating and fooling the fusion center will either deny some legitimate users from using a free band or allow users to use a band that is already occupied. Similar problems are found with distributed CRN at the time of spectrum sensing decision. Thus, it is being considered that SSDF attack could be more harmful in a distributed CRN because the false information can propagate quickly with no means to control them. While in the centralized CRNs, the fusion center can control and lessen the effect of false information by comparing the data received from all CRs.

2.4.2 Primary User Emulation Attack

Distinguishing the primary user signals from the secondary users signals is a major technical challenge that is associated with the spectrum sensing in CRs. In CRN, primary users has highest priority in accessing the frequency channel. If a primary user starts to transmit in a frequency channel occupied by a SU, the SU has to vacate the channel immediately to avoid interference to the PU. Conversely, when there is no primary user activity present within a frequency range, all the SUs has equal rights to the Spectrum Hole. Based on these paradigms, there exists the potential for malicious

secondary users to mimic the spectral characteristics of the primary users in order to gain priority access to the wireless channels occupied by other secondary users. This scenario is referred as Primary User Emulation, which is carried out by a malicious secondary user emulating a primary user or masquerading itself as a primary user. As a result, the attacker is able to have the bands of a spectrum. In the presence of energy detection, a secondary user can recognize the signal of other secondary users but cannot recognize the signal of primary users. When a signal is recognized, which is detected when a secondary user is on, it is assumed that the signal is that of a secondary user only; otherwise it concludes that the signal is of a primary user.

2.4.3 Control Channel Saturation DoS Attack (CCSD)

This attack leaves the CRN with near-zero throughput. In a multi-hop CRN, CRs communicate with each other performing a channel negotiation process. MAC control frames are exchanged to reserve channel during the negotiation phase. The common control channel has limited capacity for supporting concurrent data channels. When many CRs communicate at the same time, the channel becomes a bottleneck. The attackers take advantage of this situation and generate forged MAC control frames for saturating the channel, thus decreasing the network performance.

2.4.4 Sinkhole Attack

In a Sinkhole Attack, an attacker advertises itself as having the best route to a specific destination. The neighbouring node uses it to forward their packets. Then it can modify or drop the packets that pass through it. Another attack can be performed by an attacker known as selective forwarding, where an attacker can modify or discard packets from any node in the network. The Sinkhole attack is very effective in infrastructure and mesh architecture as all traffic goes through a base station.

2.5 Importance of Spectrum Sensing & Sharing

From the above text it is clear that the Spectrum Sensing and Sharing operations of the CRs has a very important role in Spectrum Decision making function of the CRs.

Any attack on the spectrum hole data will lead to a wrong decision by the devices which leads in selecting wrong frequency channel for communication. Spectrum Hole data can be affected in the following ways to carryout a successful SSDF attack on CR. Addressing these issues will help in preventing SSDF attack on the CRN.

- Modify the Spectrum hole data stored in the distributed devices - Solution - Maintain integrity of the Spectrum data.
- Share false data with the CRs to make wrong Spectrum Decisions - Solution - Verify/Validate the data being shared by the devices.

2.6 Blockchain

Blockchain as a technology was invented by Satoshi Nakamoto wherein he proposed a new technology that will avoid the inherent disadvantages of the Centralized record keeping and single point of failure model. The best example for this is a bank where many account holders maintain their accounts. Details and balances of the customers are maintained by the bank and the same can be reconciled by the customers. Any transaction of a customer has to be done through the centralized system of the bank. This system is based on a trust model, where all the customers trusted the bank and entrusted them with to maintain their transactions. In this scenario, any large scale Cyber attack on the bank servers will lead to alteration of records and balances of customers. Also any insider in the bank can also change the records to suit their interests. To avoid the single point failure and common trust model of the system, Satoshi proposed a model which is distributed and decentralized [4]. An illustration of the same is given below in Fig. 2.2. Here the transaction between Alice and Bob is being checked by themselves instead of a third party like a bank. These transactions once added to the block are nearly impossible to reverse but the transaction held with a bank can be easily by the bank. This system largely reduces the transfer costs and other infrastructure requirements as in case of a centralized bank. Blockchain is decentralized and distributed. All current and future nodes can come back to check whether every transaction follows the agreed rules. Apart from financial sector, blockchains can be used in most of the business models where the single point or trust based system is in place.

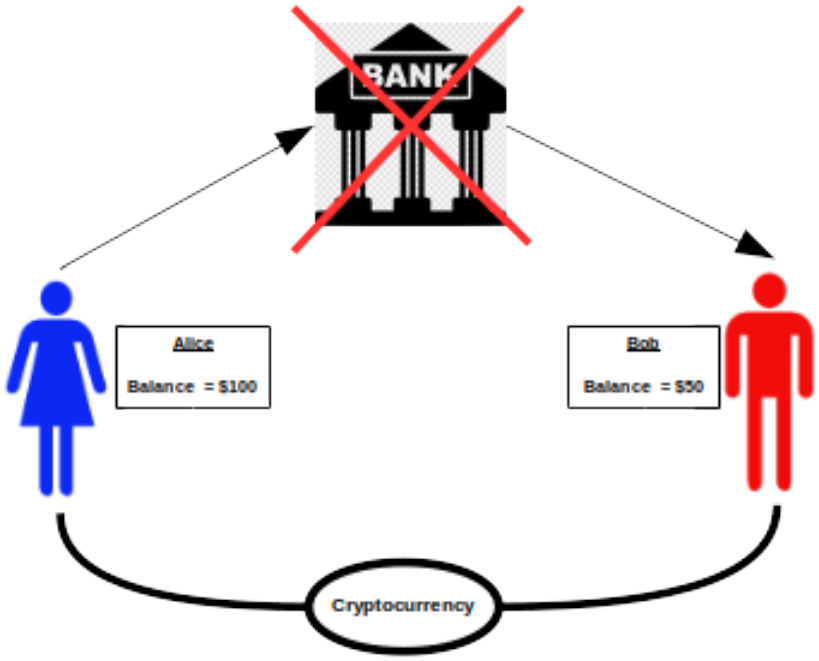

Figure 2.2: Cryptocurrency

Blockchain is a decentralized and distributed database that maintains continuously growing list of records, called Blocks. Each block contains a large number of records and transactions alongwith the timestamp and a link to the previous block. Blockchains are inherently resistant to modification of data inside the blocks due to the Hashing and chaining of the blocks. Blockchain can serve as an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way.

2.6.1 Working of Blockchain

Blockchain works as a distributed model, hence it necessitates the devices participating in the operations to be connected among each other. For example, if Alice wants to transfer $10 to his friend Bob who is also an user in Blockchain. Firstly, she will create a transaction with details, namely, the amount, debit account and credit account. This transaction is broadcast to all miners of the Blockchain network. The network has entities called "miners" which can commit this data to the blockchain. Before committing the transaction, all miners will verify the transaction. Verification is done in many ways which will depend on the type of blockchain, in this case it would be like checking whether Alice is allowed to do this transaction, does Alice has adequate balance,

etc. Once the transaction is approved, the miners compete among themselves to commit the block into the chain. This competition is called Proof of Work wherein, a difficult computational problem is solved by the miners and the winner will get the chance to commit the block into the chain. The computational problem is designed in such a way that finding a solution to it is computationally difficult however, the verification of the result is computationally easier. The node that wins this will commit the block to its chain and broadcast the same to all nodes. The Proof of Work is not the only way for the miners to win the competition. There are other works like Proof of Stake, Proof of Authority, Proof of Importance, etc. Finally the transaction amount is credited to Bob's account. The working of the same is given in Fig. 2.3

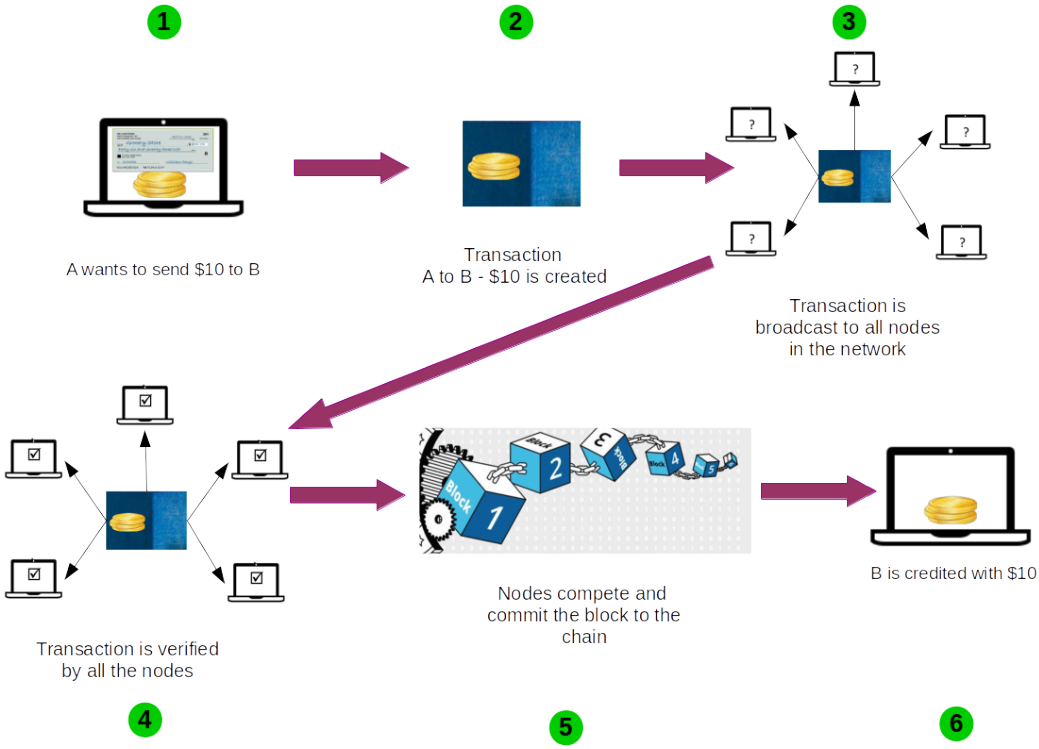

Figure 2.3: Working of a Simple Blockchain

2.7 Blockchain Terminologies

The most commonly used terminologies in Blockchain are explained in this section.

2.7.1 Block

A Block is a basic entity in the Blockchain where the transactions and other details are stored. Every Block will have a set of transactions that are verified by the miners. These transactions can be any data related to the blockchain which can be verified by the miners. The block header consists of the details related to the block. Fields of a block header is given below.

- **Version** - A version number to track software / protocol upgrades
- **Previous Block Hash** - A reference to the hash of the previous(parent) block in the chain.
- **Merkle Root** - A hash of the root of the Merkle tree of this block's transactions.
- **Timestamp** - Approximate creation time of this block
- **Difficulty Target** - The Proof-of-Work difficulty target for this block
- **Nonce** - A counter used for the Proof-of-Work algorithm.

First block of the blockchain is called the **Genesis block**. Each block is linked to its previous block by means of hash/digest of its parent block. Representative image of a blockchain is shown in Fig. 2.4. The data in a block is almost same for all the miners that are trying to commit the block. But Nonce field data is changed by the miners to fit the hash result that will satisfy the Proof-of-Work problem. The first miner to find the correct solution will get rewarded for adding the block to the chain. In case of Bitcoin, the reward in terms of Bitcoins is credited to the miner's account.

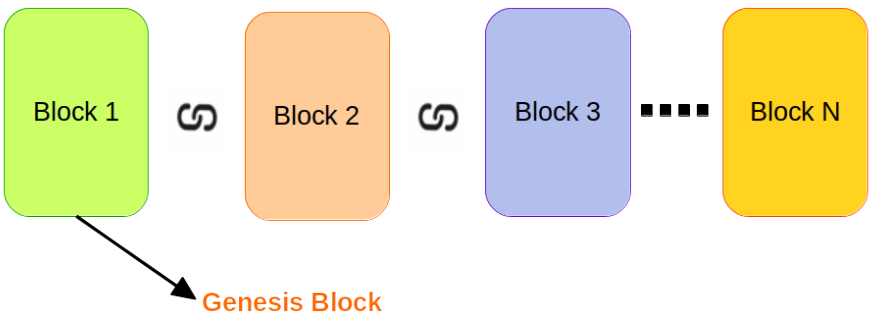

Figure 2.4: Blockchain Representation

2.7.2 Node

A node in a Blockchain is a server that stores, updates, and broadcasts full copy of the blockchain. These nodes are the distributed storage for the Blockchain. There can be any number of nodes in a Blockchain and without nodes the Blockchain will become offline and cannot be used or updated.

2.7.3 Merkle Tree

A hash tree or Merkle tree is a tree in which every leaf node is labelled with the cryptographic hash of a data block, and every non-leaf node is labelled with the cryptographic hash in the labels of its child nodes. Hash trees allow efficient and secure verification of the contents of large data structures. Hash trees are a generalization of hash lists and hash chains [11]. Merkle tree is used to reduce to re-transmission of the entire message instead only the child branch that is affected will be transmitted.

2.7.4 Consensus Algorithm

Consensus Algorithm is a procedure in which all peers of the Blockchain participate to decide the miner that will commit a block into the chain thereby reaching a consensus among the participants. In a distributed network, absence of a coordinator node necessitates the peers to decide upon a mechanism that will ensure only a single node updates the chain. This mechanism known as Consensus Algorithm ensures that the block added to the chain is unique and is created by a particular peer.

2.7.5 Proof of Work

PoW was the first Consensus Algorithm developed. In this, the miners compete among themselves to produce a hash of the next block of the chain by repeatedly performing hashes. These final hash must match with the difficulty target. Eg. leading five 0s in the hash, trailing ten 1s, etc. This is achieved by changing the nonce field of the block header. The miner with higher computational capability will succeed in proving its work and thereby committing its own version of block to the chain. There are other

consensus algorithm like Proof of Stake, Proof of Authority, Proof of Importance, etc.

2.7.6 Smart Contract

Smart contracts are computer programs that are stored on a blockchain system and automatically executed when predetermined terms and conditions are met. They are programs that run as they have been set up to run by the people who developed them. The benefits of smart contracts are most apparent in business collaborations, in which they are typically used to enforce some type of agreement so that all participants can be certain of the outcome without an intermediary's involvement. For example, if dealer has to be paid by the customer for purchase of a car, then the dealer must hold that model, transfer it to the customers name and submit the papers. Upon meeting these conditions, the smart contracts in the blockchain will automatically debit the customer's account and credit it into the dealers account. Here the involvement of peers or nodes are avoided to avoid disputes.

2.8 Blockchain in Related Works

Blockchains have been used in applications like Cryptocurrencies [4] to provide integrity to transaction based data. Though blockchains have been proposed to be used in CRNs [5] also, their use has been restricted to leasing and accessing available channels by the SUs. In [5], each SU is provided with a local currency called *Specoins* which is used to lease spectrum from a PU which holds the license for that spectrum. A PU grants lease of a *Spectrum* to the requesting SUs based on an auction mechanism (in *Specoins*) where the highest bidder will be awarded access to the spectrum for a specified duration. All these transactions between SUs and PUs in *Specoins* are recorded in blocks to maintain integrity, which will also be validated by miner SUs. The miners are rewarded for their mining work with *Specoins*. *Specoins* can be earned by SUs in two ways, 1) Mining activity i.e, successfully solving the hash problem and getting rewarded and 2) Convert a real currency to *Specoins*. There is also an option for the SUs to convert the *Specoins* to real currency. Here in this work, the Blockchain have been used as cryptocurrency implementation where the currency transactions are stored in the chain and blockchain infrastructure used to maintain the chain. Using this mechanism

in a memory constrained devices like IoBT is not proven. Blockchains have also been proposed to be used in Wireless Sensor Networks (WSN) using Blockchain Authentication and Trusted Model (BATM) [7][8] to provide an alternate to the PKI architecture to store private and public keys.

However, we propose a Proactive Blockchain based Spectrum Sharing Protocol (ProBLeSS) to provide security to CR based IoBT networks against SSDF DoS attacks without using the resource intensive key management architectures. In this, we propose to use the blockchain to store the spectrum availability information as transactions in blocks. The main contribution of this work is that it leverages the two important feature in a blockchain architecture for providing security to CR-IoBT networks against SSDF attacks are :

- A robust *Consensus Protocol* to provide security to spectrum sharing data.
- A *Smart Contract* for validation of spectrum data.

CHAPTER 3

ProBLess, a Solution for SSDF Attack

3.1 The Proposed Algorithm

CR as a technology was proposed by Mitola [2] for opportunistic use of the less used portions of the radio spectrum using intelligent devices which will be using *Software Defined Radios* (SDR). A cognitive radio can change its transmitter parameters based on the interaction with the environment in which it operates. It has the ability (cognitive capability) to sense and gather information (such as the transmission frequency, bandwidth, power, modulation, etc) from the surrounding environment as well as has the ability (re-configurability) to swiftly adapt the operational parameters, for optimal performance, according to the information sensed. It is an intelligent wireless communication system that is aware of its surrounding environment and uses the methodology of understanding-by-building to learn from the environment and adapt its internal states to statistical variations in the incoming RF (radio frequency) stimuli by making corresponding changes in certain operating parameters in real-time. *Primary User* (PU) in the CR are the devices that holds the license to communicate in the channel. *Secondary User* (SU) are the CRs that opportunistically uses the free channel for communication.

CRs enable the usage of temporarily unused spectrum, referred to as *spectrum hole* or *white* space, and if a PU intends to use this band, then the SU should seamlessly move to another spectrum hole or stay in the same band, altering its transmission power level or modulation scheme to avoid interfering with the primary user. Traditional spectrum allocation schemes and spectrum access protocols may no longer be applicable when secondary unlicensed users coexist with primary licensed users. If secondary users are allowed to transmit data along with primary users, the transmissions should not interfere with each other beyond a threshold. On the other hand, if secondary users can transmit only in the absence of primary users, then a secondary user transmitting data in the absence of a primary user should be able to detect the reappearance of the primary user and vacate the band.

Blockchains inherently provide the security from tampering of data i.e. integrity, by its architecture. However, in a broadcast domain like the CRNs, verifying the authenticity of a sender possesses major challenges [10]. Use of symmetric keys or PKI has been suggested in the past to solve the authentication problem. However, issues like key distribution and key management becomes challenging in a resource constrained ad-hoc network based architecture with dynamically varying users such as CR-IoBTs. Hence, we need a key-less protocol that can provide a certain assurance that even if the authenticity of any joining battlefield users is not ascertained, the malicious BEs will not be able to influence the spectrum decision to such an extent that it is not acceptable for effective communication. In this regards, ProBLeSS provides one such solution using Blockchain architecture.

3.2 Blockchain Data Structure for ProBLeSS

Traditional Blockchain infrastructure has high requirement for computational and power. The power and computational resources availability in IoBT devices are very limited. The Consensus Algorithm (Proof-of-Work) necessitates high computational power from the Miners, however, ProBLeSS has suggested a new mechanism where the Consensus is based on a simple mathematical calculation in which the values are derived based on the feedback from the devices in the network. Accordingly, the Blockchain infrastructure required for implementing ProBLeSS algorithm is designed. The blocks of this blockchain will have four fields as given below.

- **Previous Hash** - Hash of the previous/parent block
- **Transaction Root** - Merkle-tree Root of the transaction (Spectrum Hole) data that are to be committed in the block.
- **Time Stamp** - Time of the Day
- **Seed** - Identity of the CR-IoBT device that is committing the block to the chain.

According to PROLEMus [1] algorithm, the proactive learning system of a CR needs the transaction (Spectrum Hole) data of the CRs for past 70 seconds at the spectrum sensing rate of 100ms to make a most efficient spectrum decision. In a traditional

blockchain system, the transactions of a block are stored alongwith the block in the chain, which will lead to a huge overhead in terms of memory requirement for the IoBT devices. Since the algorithms like PROLEMus require only certain amount of historical data, the remaining data can be ignored for decision making. Accordingly, to reduce the amount of storage memory required for blocks in an IoBT device, the block in ProB-LeSS will have only the Root Hash of the Merkle-Tree of transactions. Transactions of the block can be stored separately by the device suiting to its memory availability. This design will largely reduce the memory requirement to store the blockchain in an IoBT device wherein the blocks are of fixed size. When a device receives the transactions data from other device, the integrity of the data can be verified with the Merkle-Tree Root hash in the block and Merkle Tree has its own advantage of reducing re transmission of the whole file again pending verification. Detailed Data structure of the ProBLeSS algorithm is given in Fig 3.1

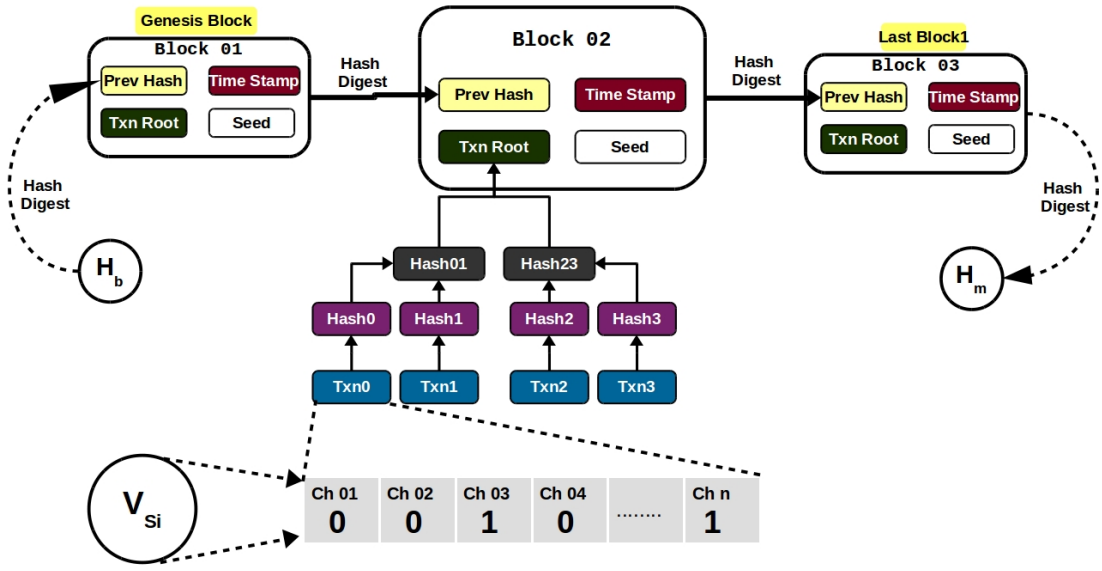

Figure 3.1: ProBLeSS Blockchain

3.3 System Architecture

A CR based IoBT consists of *n* PUs who have licensed frequency for communication. Each PU (P_i , $1 \leq i \leq$ n) has one dedicated channel (C_i , $1 \leq i \leq$ n). The ad-hoc CR-IoBT network consists of *m* BEs (S_i , $1 \leq i \leq$ m) who will use one of the channels dedicated for PUs (C_i), without interfering their communication, whenever it is free.

Every BE holds a blockchain that stores the spectrum sensing data. The BEs compete to add their block to the blockchain and share it with all other BEs. There exist a requirement to select a BE for adding a block of spectrum sensing data to the blockchain and synchronize the chain so that each BE can have a similar copy of the chain. A consensus mechanism called the ProBLeSS Consensus Algorithm (PCA) is used to add a block to the ProBLeSS blockchain. A mechanism similar to *longest chain match* [4] is used to synchronise the blockchains among all BEs. In addition, each transaction in a newly added block is vlaidated using a smart contract called the ProBLeSS Smart Contract(PSC).

3.4 Attacker Model

A CR-IoBT network consists of BEs who opportunistically use the licensed frequencies of the PUs to transmit data. However, one of the common Denial of Service (DoS) attacks on such a CRN is the SSDF attack. In the SSDF attack, one of the BEs generates false spectrum data to influence the decision mechanism and force other BEs to transmit on busy channels thereby degrading performance in terms of increased backoffs and reduced channel utilization [1]. In our architecture, we assume that a malicious BE will receive the spectrum data from all other BEs and try to add its malicious block in the blockchain. This block will be used by other BEs to take decision resulting in degraded performance. In addition, one of the BEs can also be captured by the malicious user which can lead to compromise of any setup parameters etc. We also assume that the malicious BE will have large resources in terms of computation and storage as compared to the resource constrained IoBT BEs.

3.5 ProBLeSS Consensus Mechanism

The description of a block in ProBLeSS blockchain is as given in Fig. 3.1 where txn0, txn1, etc depict the transaction data (spectrum data that indicates availability of the channels) that is stored in a Merkel Tree with the root as *Txn_Root*. The hash of the previous block is added as *Prev_Hash* in a new block and is also called as the *Master_Hash*(H_m). Every BE is provided with a base hash H_b that is used as the *Prev_Hash*

in the genesis block of the blockchain. The block also contains the *Timestamp* giving the time of block creation. It is worth noting that ProBLeSS can reject a block based on its timestamp to avoid adding very old spectrum data to the blockchain that may be used for spectrum decision. The value of *seed* is an unique identifier that gives identity of the BE that added the block. The use of *seed* will be explained in later section while discussing the ProBLeSS consensus mechanism.

In blockchain, a transaction is added only after it has been approved using a *Consensus Protocol*, which ensures it is the only correct version of data. To implement consensus based on Proof of Stakes paradigm [9], we introduce a *Winner Incentive* concept where the reputation of each user that adds a block to the blockchain is updated based on the quality of data being added in that block. This quality is determined by feedback from users that use that data for making spectrum decisions. These features ensures that only good quality data is added to the blockchain because malicious users adding false data are penalised in their reputation and prevented to add further blocks. We implement the ProBLESS consensus mechanism using the ProBLeSS Consensus Algorithm (PCA) and PSC as given at Algo 1

3.6 PCA Algorithm

ProBLeSS Consensus Algorithm (PCA) (refer Algorithm 1) will be invoked at regular intervals in a BE. At each sensing interval, every BE S_i ($1 \leq i \leq m$ and ID = i) senses all the channels c_j ($1 \leq j \leq$ n) and generates a *Spectrum Data Vector* V_{S_i} that gives the free/busy state of each channel c_j ($1 \leq j \leq$ n). If a channel c_j (say) is free then V_{S_i} [j] is set as 1, else 0. In addition, every BE maintains a *Count* (n_{S_i}, $1 \leq i \leq$ m) which gives the cumulative count of the received vectors V_{S_i}s received in the past PCA *epoch*. Thereafter, every BE transmits its sensing data V_{S_i} to all other BEs. In this process, each BE obtains the sensing data from all other BEs in form of the V_{S_i}s that depicts a transaction data in the blockchain. The BEs use the transaction data to generate a block for adding to the blockchain.

After the BEs generate a block, they broadcast their updated n_{S_i} to all other BEs. Each BE also maintains another count called the WinCount (*WinCount*$_i$) that keeps the count of the number of blocks it added in last PCA $epoch_{CA}$. The *WinCount*$_i$ is also

Algorithm 1: ProBLeSS Consensus Algorithm

Input: *Spectrum Data Vectors, BE ID and PCA Values*
Output: *Updated Blockchain* with new added block of spectrum data vectors

1 **for** each *Spectrum Data Vectors* received from other BEs **do**
2 **if** *BE is marked as malicious* **then**
3 Validate Spectrum Data Vector using PSC
4 **if** *Valid Spectrum Data* **then**
5 Increase *Count* of Spectrum Data Vectors
6 break if PCA $epoch_{CA}$
7 Generate a Block using *Spectrum Data Vectors* as *transactions, Previous hash, Time Stamp and ID*;
8 Broadcast own Count to all other BEs and update *Count* of all other BEs
9 Find the BE with maximum [*Reputation/Count*] and declare it as *Winner*.
10 **if** I am the *Winner* **then**
11 Increase *WinCount*;
12 Add the new *Block* of transaction to own copy of blockchain;
13 Share the length of own blockchain with all other BEs.
14 **if** I have the longest chain **then**
15 Share the new block with all other BEs.
16 **else**
17 Get the new block from *Winner* BE.
18 **for** *each other BE* **do**
19 *Decision Parameter = Backoff Rate / Total Channels*
20 *Reputation = [Reputation + [1 - Decision Parameter]]/2 + [WinCount / Total Number of BEs]*
21 **if** Reputation *is less than a threshold* **then**
22 mark BE as malicious and inform all
23 **if** *PSC* $epoch_{SC}$ **then**
24 Validate Spectrum Data Vectors using PSC
25 Reset the *Count* and *WinCount*of all BEs
26 GOTO Line 1.

broadcast to all other BEs along with n_{S_i}. In addition to $winCount_i$, the BEs also use a value called the *Decision Parameter* (D_{S_i}). The D_{S_i} of each BE is calculated from the average number of backoffs per second (*Backoff Rate*) faced by other BEs that used its added block for making spectrum decision. The identity of the BE who added a block is found from the value of *seed*. Thus, after every successful transmission by a BE, it shares the *Backoff Rate*(B_{S_i}) with all other BEs for updating the D_{S_i} of BEs whose blocks were used for making the spectrum decision. Note that the D_{S_i} improves on addition of blocks with good spectrum data and suffers otherwise. Thus, each BE uses the five values (*PCA Values*) of each other BE (n_{S_i}, *WinCount*$_i$, D_{S_i}, B_{S_i} and r_{S_i}) for selecting the *Winner* BE. The BE with the maximum [*Reputation/Count*] value is chosen as the *Winner* BE that adds its block to the chain and shares with others.

The PCA uses a mechanism similar to the longest chain match protocol to share the longest copy of the blockchain between the BEs. Note that, the longest chain will always be with the BE that emerges as the *Winner* at every invocation of PCA. Let us prove this by contradiction. Consider that a BE has emerged as the *Winner* and it does not have the longest chain after adding its current block. This means that in the previous invocation of PCA, there exists a BE whose chain was longer than the *Winner* BE. This is not possible because all BEs synchronize their chain after each invocation of PCA. During the synchronization process, the *Winner* BE broadcast its copy of blockchain along with the H_m. Finally, after synchronizing the blockchain, every BE updates the *Reputation* of all other BEs using the updated *PCA Values* if its *Reputation* is less than a *threshold* (say 0.5).

3.7 ProBLeSS Smart Contract

ProBLeSS invokes the PSC in every BE for validating the transactions in the blockchain after regular intervals (PSC $epoch_{SC}$). In addition, the PSC validates each V_{S_i} received from a BE after it has been marked as malicious. On invocation, BEs compare each channel availability (free/busy status) with the average channel availability of all the transactions in the last 7 blocks (blocks used for spectrum decisions by PROLEMus [1]) in its copy of the chain. The average channel availability is calculated by considering the channel to be on an average free/busy if more than half of the transaction data

declare it as free/busy. The transactions in which more than half of the channels match the average channel availability are confirmed and others are discarded. Thus every BE maintains a blockchain with spectrum sensing data that is validated by itself. The ProBLeSS Smart Contract ensures that even a malicious user manages to add a block of false data and does, its transactions are subsequently discarded by validation using PSC.

3.8 ProBLeSS Security

Security of a CR-IoBT depends on two main functions; 1) avoiding the malicious BEs to tamper with the old data; and, 2) avoid the malicious BE to generate false data for influencing the decision. We propose the blockchain architecture for CR-IoBT networks that inherently provides security against tampering of stored data[4]. In addition, we propose a mechanism where malicious users are prevented from adding blocks of false data into the ProBLeSS blockchain that achieves similar security against the SSDF attack, as would have been achieved using BE authentication using a *Winner Incentive Looser Penalty* concept, similar to bitcoins blockchain [4]. The notion of incentive is achieved by increase in performance of the *Winner* BE that quickly adds the block to its blockchain and starts transmission when other BEs wait for receiving the newly added block from it. This provides the *Winner* BE additional time and available channels for transmission. The notion of penalty is achieved by preventing a single BE to continuously adding its transactions in the blockchain. ProBLeSS ensures that even if malicious users are part of the network, they cannot influence false decisions by always adding their false spectrum data to the blockchain because it will hamper their *Reputation* as well as fails the validation. Hence, every BE who is part of the CR-IoBT strives to win the contest making the PCA and PSC susceptible to SSDF only on 51% attack [4].

CHAPTER 4

IMPLEMENTATION

The proposed ProBLeSS algorithm has to be implemented and checked for feasibility in a device that has low hardware resources that are similar to CR devices. The implementation and evaluation of results could be done in two steps. Firstly, to implement it in limited number of physical devices and check the parameters. Secondly, simulate it in a CR simulator with more devices and evaluate the results. With the feasibility check ready for the algorithm, it can be evaluated in a simulator for its efficiency under SSDF attack. The Ubimote-SEZ device, selected for feasibility check of the algorithm, is a Wireless Sensor Network (WSN) device. It is assumed that this device can be attached to a CR device for all Spectrum Sharing activities wherein all other CR related activities are carried out by the CR device themselves. Accordingly, the feasibility of the algorithm was checked in Ubimote-SEZ device and performance parameters were checked using a CR simulator with limited capability.

4.1 C-Mote (CDAC)

Initially a device(c-Mote) was readily available in the laboratory to carry out a feasibility study of the proposed protocol. The device was running on MSP-430 processor with peripheral devices support. However, the hardware is of old model and doesn't support advanced Hashing and Encryption algorithm standards. Although, the device was not used for the current work, it was used to gain hands-on knowledge on programming similar devices.

CDAC, Bengaluru was approached for a device that can support the latest hashing and encryption algorithms alongwith advanced hardware features. Accordingly, they have provided us with a device (Ubimote-SEZ) designed and developed by them.

4.2 Ubimote-SEZ

Ubimote-SEZ is a wireless Sensor Node designed with a robust RF front end with state of the art Security engines and efficient power management system. This device also supports many features which are suitable for Wireless Sensor Network indoor and outdoor deployments. The main components of the device are

- TI CC2538, an ARM CortexM3 with 32 KB on-chip RAM and 512 KB on-chip flash with a robust IEEE 802.15.4 radio.
- Range extender with LNA and PA.
- SMA connector for interfacing 2.4 GHz SMA antenna
- 8Mbit Serial Flash memory for additional storage.
- JTAG Connector for programming and debugging
- USB interface for peripheral functionality.
- Power through USB Connector and External Battery

The device is powered by CC2538 SoC by Texas Instruments. The SoC is an ARM cortexM3 chip with 32KB on-chip RAM and 512 KB on-chip flash with a robust IEEE 802.15.4 radio. In addition to this, the device has 8Mb serial flash memory for additional storage, JTAG connector for peripheral functionality and external power options. Jlink EDU is the debugger being used to flash the compiled binary/object file into the device's flash memory.

The CC2538 chip has the following features:-

- ARM Cortex-M3 processor core - 32-MHz operation, SysTick timer
- On-chip memory
  - Up to 512KB single-cycle flash memory up to 16 MHz; a prefetch buffer improves performance above 16 MHz
  - Up to 32KB single-cycle SRAM

  - Boot loader and utility library in ROM

- Advanced serial integration
  - USB 2.0 full-speed (FS) device (12 Mbps)
  - Two Universal Asynchronous Receiver/Transmitters UARTs
  - One inter-integrated circuit (I2C) module
  - Two Synchronous Serial Interface modules (SSIs)

- System integration
  - Direct memory access (DMA) controller
  - System control and clocks including on-chip 16-MHz oscillator and 32-MHz crystal oscillator
  - Four 32-bit timers (up to eight 16-bit)
  - 32-bit, 32-kHz sleep timer
  - 32 GPIOs
  - Fully flexible pin muxing allows use as GPIO or any peripheral function

Hardware and Software Requirements to setup programming of the device is given in Appendix A. Image of Ubimote-SEZ with the details of important ports is given in Fig. 4.1

4.3 Hands-on on the Device

While the study on SSDF attacks in CRN and development of the proposed protocol (ProBLeSS) was going on, concurrently, hands-on programming and utilizing the device for implementing the proposed algorithm was carried out. The following features of the device has been successfully programmed and tested.

1. Advanced Encryption Engine (AES) Electronic Code Book (ECB) mode operation
2. SHA-256 hash generation

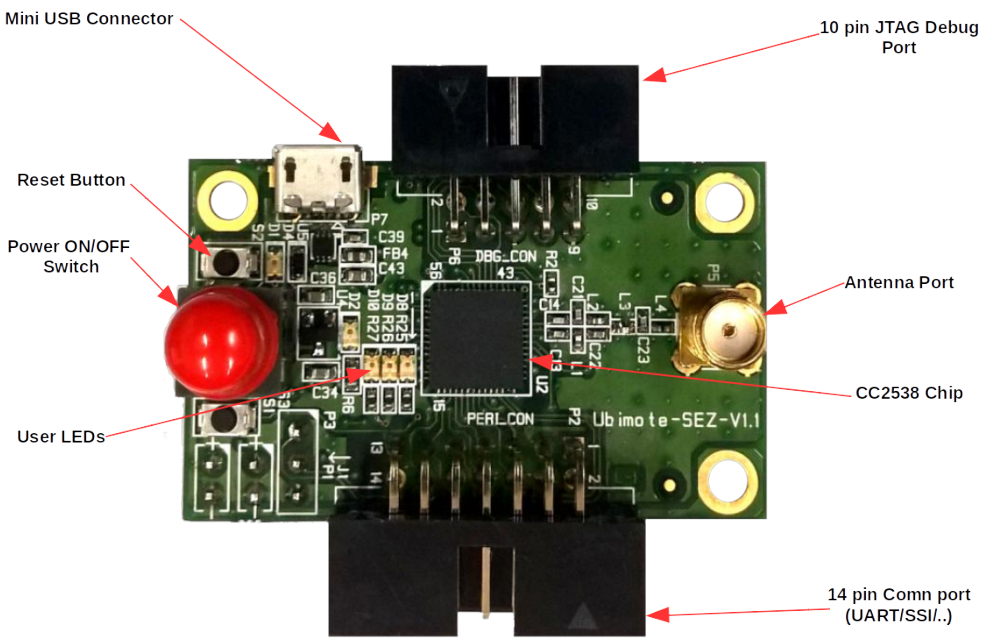

Figure 4.1: Ubimote-SEZ

3. Basic RF transmit and receive
4. General Purpose Timers
5. Systick timer

4.3.1 AES (ECB) Mode operation

AES engine of the device has support for 128, 196 and 256 bit keys. It provides the following functionality:-

(a) CCM, GCM, CTR, CBC-MAC, ECB modes of operation

(b) SHA-256 hash function

(c) Secure key storage memory

(d) Elliptic Curve Cryptography (ECC) and RSA-2048

The device supports Advanced Encryption Standard (AES) in ECB, CBC, CTR, GCM, CCM and CBC-MAC modes of operation. In ECB mode, a block of data is given to the AES engine along with a symmetric key to get the corresponding ciphertext. The input data given to the module was varied from 16 bytes to 12288 bytes to find the time taken

by the module to encrypt the data. The time taken was calculated by counting the clock cycles that have taken to complete the operation. Since the clock was set at 32MHz frequency, the time per clock cycle was calculated to 0.3125ns. Later the total cycles were multiplied with the time per cycle to get the total time taken for the operation. Timer modules were used to find the time taken for each operation. The details of the plain text data length given to the engine and the time taken by it are shown in TABLE 4.1

Input Data (in Bytes)	Time Per Bit = Time taken / (#bytes * 8) (in ns)
16	4.1015625
32	3.0895996094
48	2.7408854167
64	2.5665283203
80	2.4619140625
96	2.392171224
112	2.3423549107
128	2.3049926758
256	2.1643066406
512	2.0989990234
1024	2.0663452148
2048	2.0500183105
4096	2.0736885071
8192	2.0695590973
12288	2.0681826274

Table 4.1: AES (ECB) Encryption Timing Measurement

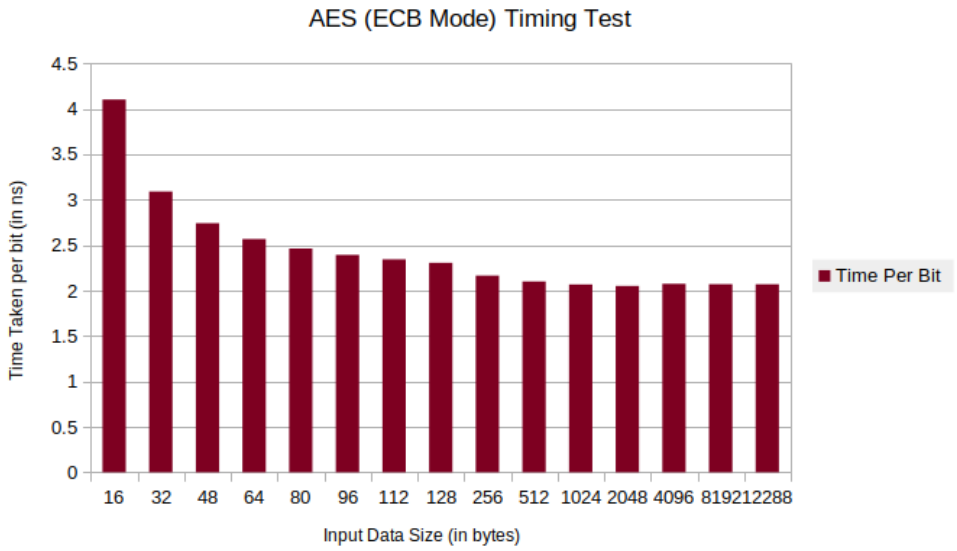

Figure 4.2: AES (ECB) Timing Measurement

The timing measurement chart of AES operation (Fig 4.2) tells us that the time taken for encryption of a data shorter than 64 bytes will take an average of 3ns per bit. However, when the size of the input data increases, the time taken for per bit encryption settles down at around 2ns per bit. This is an important observation for our work, as it will decide upon the time taken to encrypt the data before transmitting it through the radio.

4.3.2 SHA-256 hash generation

Secure Hashing Algorithm of digest size 256 bits is the default hashing algorithm in the device. AES engine is being used to perform the hashing operation. Hence at a given time, only one operation (either SHA or AES) can be performed by the AES engine. The following functions were written to generate digest for a given input.

(a) uint8_t sha256_compute (uint8_t * input_data, uint32_t data_len, uint8_t * final_hash)

(b) void print_sha256_hash (const uint8_t * computed_hash, uint32_t length)

The parameters to the above functions are self explanatory. The programs were specifically run for various lengths of the input data and repeatedly run for number of iterations. The results of the same are given in TABLE 4.2

	Iterations							
Bytes	**1**	**10**	**50**	**100**	**200**	**500**	**750**	**1000**
1	24.922	24.840	24.749	30.365	30.358	24.768	24.767	24.767
50	1.070	1.069	1.068	1.067	1.067	1.066	1.066	1.066
100	0.970	0.943	0.942	0.942	0.942	0.969	0.969	0.969
500	0.761	0.810	0.810	0.810	0.810	0.761	0.761	0.761
1000	0.740	0.803	0.803	0.803	0.803	0.739	0.739	0.739
2500	0.727	0.799	0.799	0.799	0.799	0.727	0.727	0.727
5000	0.794	0.721	0.721	0.721	0.721	0.793	0.793	0.793
7500	0.792	0.719	0.719	0.719	0.719	0.792	0.792	0.792
10000	0.791	0.718	0.718	0.718	0.718	0.791	0.791	0.791
15000	0.790	0.717	0.717	0.717	0.717	0.790	0.790	0.790
20000	0.789	0.716	0.716	0.716	0.716	0.789	0.789	0.789
30000	0.789	0.716	0.716	0.716	0.716	0.789	0.789	0.789

Table 4.2: SHA 256 Hash Timing Measurement

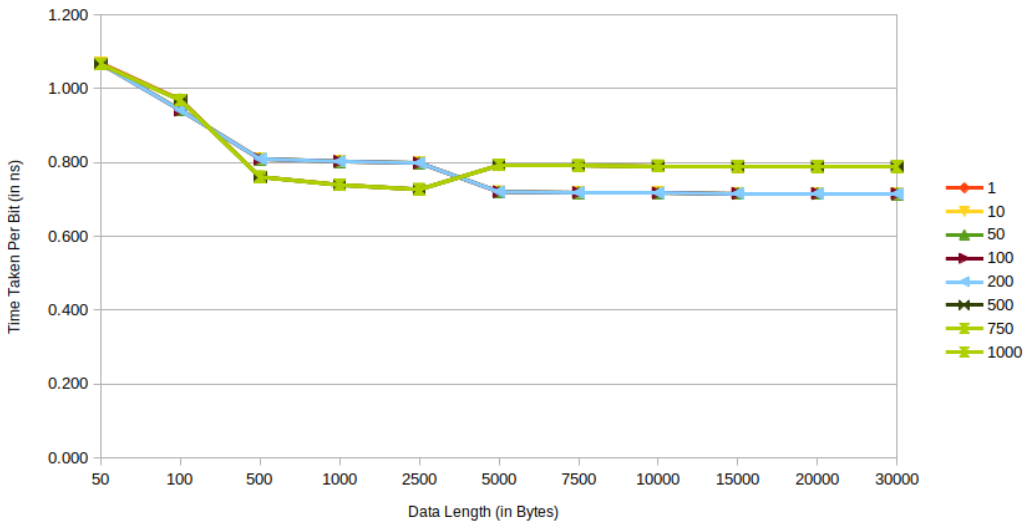

Figure 4.3: SHA-256(128 bit) Timing Measurement

In Fig 4.3, the coloured lines represent the total number of iterations a particular hashing was run. And it is clear from the chart that the time taken to hash a bit of data is around 1ns, when the size of input data is around 500 bytes. The time reduces further and stabilizes around 0.78 to 0.80 ns for large input data.

4.3.3 Radio Transmit / Receive

The Ubimote-SEZ device provides a highly integrated low-power IEEE 802.15.4-compliant radio transceiver. The radio subsystem provides an interface between the MCU and the radio which makes it possible to issue commands, read status, and automate and sequence radio events. The radio also includes a packet-filtering and address-recognition module. Any protocol like ZigBee, Thread, SNAP, etc can be programmed according to the user requirement from the 802.15.4 complaint radio interface. The radio interface is important for implementation of the ProBLeSS, as the CR-IoBTs have to communicate among them and exchange the spectrum data for collaborative spectrum sensing mode of spectrum exchange. The device was programmed to transmit and receive data through their radio interface based on timer interrupt. The radio was set in receiving mode continuously and transmit based on a periodic timer interrupt that was set for 0.5 seconds. The following functions were used for the radio transmit and Receive operations:-

(a) basicRfInit () - Initialize the RF radio module

(b) basicRfReceive() - Read the data from the receiver

(c) basicRfSendPacket() - Transmit data to the specified address/broadcast

Time taken by the radio module to transmit a bit was calculated by sending a fixed length of data through the transmitter to another device and waiting for the receipt of acknowledgement from the receiving device. The total time taken for the entire cycle was calculated by varying the data length. The maximum payload length possible for a MPDU (Layer 2) packet is 103 bytes. Hence for any data length more than the acceptable payload, the data was fragmented and sent in different packets. Details of the timing measurement for different sizes of data is given in TABLE. 4.3

Bytes Sent	**Time Taken per Bit(in ns)**
1	1398.34
2	747.11
3	530.12
4	421.57
8	258.90
16	180.90
32	138.47
64	117.27
96	110.20
128	98.60
160	89.73
192	83.82
224	79.60
253	76.69

Table 4.3: Radio Module Transmit Receive Timing Measurement

The timing measurement chart (Fig. 4.4) for radio interface clearly shows that more number of bytes transmitted in a single transmission will take less time than transmitting the same in more number of attempts.

4.3.4 General Purpose Timers

The device provides General Purpose Programmable timers which can be used to count or time external events that drive the timer input pins. Each 16- or 32-bit GPTM block provides two 16-bit timers or counters that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer. The

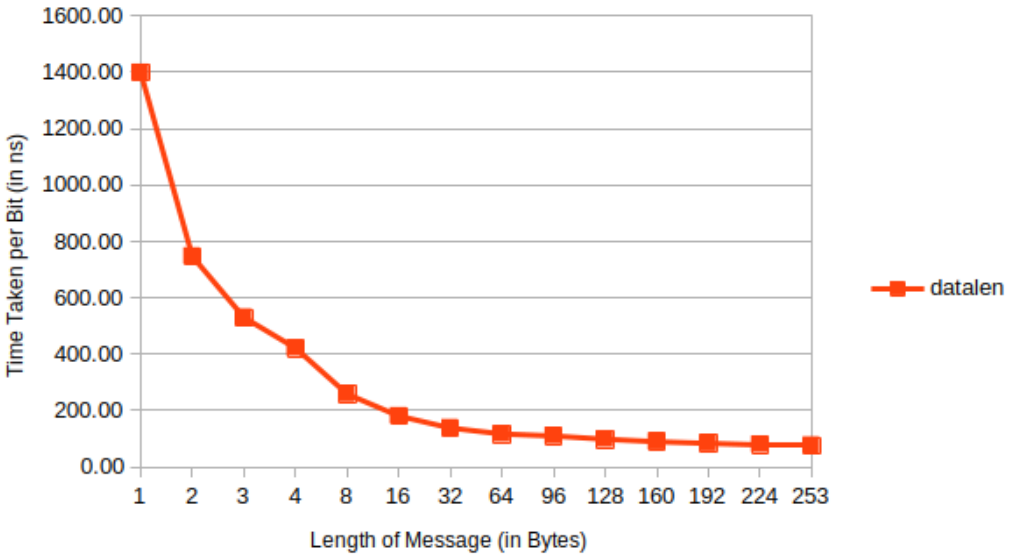

Figure 4.4: Radio Transmit/Receive Timing Measurement

general-purpose timer module (GPTM) contains four 16- or 32-bit GPTM blocks with the following functional options:

(a) 16- or 32-bit operating modes:

- 16 or 32-bit programmable one-shot timer
- 16 or 32-bit programmable periodic timer
- 16 bit general-purpose timer with an 8-bit prescaler
- 16-bit input-edge count- or time-capture modes with an 8-bit prescaler
- 16-bit PWM mode with an 8-bit prescaler and software-programmable output inversion of the PWM Signal.

(b) Count up or down.

(c) Daisy-chaining of timer modules to allow a single timer to initiate multiple timing events.

(d) Timer synchronization allows selected timers to start counting on the same clock cycle.

(e) User-enabled stalling when the microcontroller asserts CPU Halt flag during debug.

(f) Ability to determine the elapsed time between the assertion of the timer interrupt and entry into the interrupt service routine.

The timer module provides two half-width timers/counters that can be configured to operate independently as timers or event counters or to operate as a combined full-width timer. The timers provide 16-bit half-width timers and a 32-bit full-width timer. The two half-width timers provided by a timer module are referred to as TimerA and TimerB, and the full-width timer is referred to as TimerA. When configured as either a full-width or half-width timer, a timer can be set up to run as a one-shot timer or a continuous timer. If configured in one-shot mode, the timer ceases counting when it reaches 0 when counting down or the load value when counting up. If configured in continuous mode, the timer counts to 0 (counting down) or the load value (counting up), then reloads and continues counting. When configured as a full-width timer, the timer can also be configured to operate as an RTC. In this mode, the timer expects to be driven by a 32.768-KHz external clock, which is divided down to produce 1-second clock ticks.

Timing measurements of all the above modules of the device were carried out using the GPTIMER0 (GPTM Module 0) where the Timer A was configured as below:

(a) Mode : *One-Shot Operation*

(b) Timer Width : *Full-width (32 bits)*

(c) Counter Type : *Downward Count*

(d) Count Value : *1,000,000,000*

4.3.5 SysTick Timer Module

SysTick Timer module is part of the Nested Vector Interrupt Controller (NVIC) in Cortex-M microprocessors of ARM. Its intended purpose is to provide interrupt for an RTOS, but it can be used for other simple timing purposes also. Unlike the other timer interrupts, SysTick interrupt handler does not need to clear the SysTick interrupt source because the NVIC automatically does so when the SysTick interrupt handler is called. The SysTick timer can be used to handle periodic tasks of a CR-IoBT. The tasks like

transmitting the spectrum hole vector data to all the IoBTs every 10ms can be carried out with this timer. In this, the interrupt handler can use to transmit function of the device to immediately broadcast the vector to all the IoBTs. The functions used to handle SysTick Timer are given below:-

(a) SysStickEnable(), SysStickDisable()

(b) SysTickIntEnable(), SysTickIntDisable()

(c) SysTickRegister(), SysTickUnRegister()

(d) SysTickPeriodSet(),SysTickPeriodGet()

(e) SysTickValueGet()

The SysTick timer module can also be used to calculate timing measurement for a particular operation. This can be done by using the SysTickValueGet() function which will give the counter value at the particular cycle. The value of the counter can be taken before and after the particular operation to find the time taken to complete the operation.

4.4 Implementation of ProBLeSS in Ubimote-SEZ

The Ubimote SEZ device has a radio module as per IEEE 802.15.4 standard [12] for low data rate Personal Area Networks (PAN) devices. Hence it can transmit and receive in a particular frequency with Layer 2 frames as per IEEE 802.15.4 format. The device is Bare Metal with no Operating System. The programmer has to be aware of the system level details of the device to implement an algorithm. Since Ubimote-SEZ is not a Cognitive Radio device, all the cognitive radio related functions are to be simulated in the device. The details of the same are mentioned in next section. Simulation of large number of CR devices and reading the performance of the algorithm would be an ideal condition for evaluating the algorithm. However, due to this device available in the lab, it was decided to check the implementability of the algorithm in a resource constraint device like Ubimote-SEZ.

Connection setup of the device for programming is given in Fig. 4.5. More number of devices could be added to this setup with additional set of accessories with each device.

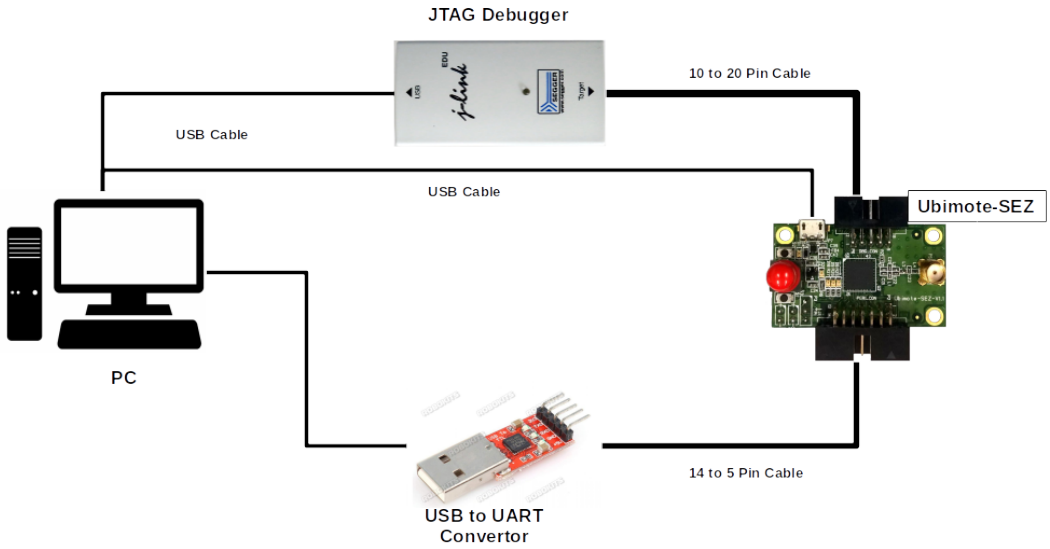

Figure 4.5: Ubimote-SEZ Programming Setup

4.4.1 CR related functions

Ubimote-SEZ is a simple Wireless Sensor Network device. Therefore, the CR related functions and other prerequisites were assumed or simulated. The list of such assumptions and simulations are given below:-

- **Number of Spectrum Channels** - The number of available channels for a CR depends on the service provided by it. Since it is a simulation, it has been assumed that eight (8) channels are available for the CR devices to sense and use. Eight channels were assumed so that the availability status of all the channels could be represented in a byte.

- **Spectrum Sensing Data** - A CR device has to sense the spectrum at a certain interval to find the Spectrum Hole data to share it among the other devices. In this program, the Spectrum Sensing data has been simulated as data of one byte where each bit represents the free/busy status of a particular channel. For example, 1 byte data, 0x11 is 00010001, implies that first and fifth channels are busy and other six channels are free.

- **Backoff Rate** - Backoff rate is an important parameter used by the algorithm to determine the malicious nature of the spectrum sensing data. Backoff rate is the rate at which a radio device backoffs while trying to transmit in a particular channel. The backoff rate for a channel is randomly generated in the implementation.

The value for backoff-rate lies between 0 to 99.

- **Common Communication Channel (CCC)** It has been assumed that all the CR devices will have a Common Communication Channel through which they can exchange the information and data. Accordingly, a single frequency was used for communication among the devices.

- **Lossless Medium** The devices were configured to communicate in a setup where the data is not lost while transmitting. The coordination among the devices were hardwired to avoid any loss of the transmitted frame.

4.4.2 Synchronization of the Devices

It was mentioned earlier that the devices will communicate among themselves using CCC which is a single frequency channel. Ubimote-SEZ has a Half-Duplex radio interface where the device can either transmit or receive at a time. It means that the devices need to coordinate among themselves while transmitting or receiving the data. At a given point of time, if a device is transmitting, then all other devices must be receiving at that time. The requirement was implemented by making a time sharing arrangement among the devices. In this, the devices are powered ON at same time and the General Purpose Timer was used to make the devices to transmit using round robin method. When one device is transmitting, other devices will learn that its not their turn and keeps listening to the CCC frequency during that period. A device knows about its turn to transmit with the help of count value of the timer interrupts.

4.4.3 Data Structures Used

Implementation of the algorithm has made use of inbuilt data structures of the C language. However, due to non-availability of operating system and layer-3 level functions, certain custom data structures were used for smooth implementation. Important ones are discussed below.

- **Spectrum Hole (Transaction) Data** - Every device in the network will broadcast the Spectrum Hole data sensed by it. The data being shared will have the device identity and the spectrum hole data. This data is used in the block as transactions.

- **Block** - The structure of the block is already defined in 3. The defined structure contains Hash of the previous block, root hash of the transactions that are related to the block, seed (identity) of the device committing the block and time.
- **Messaging Format** - The devices exchange various data among themselves in the algorithm. A message format was required at Layer 3 level for message control in the device. A standard message format with from and to address of the devices where identity of the device will be used, type of the message, payload length and the payload. The message type and their corresponding codes are given below.
  - "**0x10**" - Spectrum Hole Data.
  - "**0x20**" - Spectrum Hole (transactions) count received by each device from other devices.
  - "**0x30**" - Block related data that has only newly committed block.
  - "**0x40**" - Block related data that has only transactions of the newly committed block.
  - "**0x50**" - Backoff Rate data.

4.4.4 Implementation Details

The timers were used to coordinate various functions among the devices to implement the ProBLeSS algorithm. The device is always on radio receive mode and the timers were used to enable the flags that will carryout their respective function. The functions associated with each timer is given below.

- Timer 0A - To transmit the sensed Spectrum Hole data.
- Timer 1A - To carryout ProBLeSS Consensus Algorithm.
- Timer 2A - To carryout ProBLeSS Smart Contract Algorithm.

4.4.5 Working of ProBLeSS in Ubimote-SEZ

The programming of Ubimote-SEZ device was carried out using JTAG based debugger and a simple text editor. Configuring the device and computer for programming

is given at Appendix A. The status of the device is sent through serial data communication to the computer. The serial data could be received using the USB-UART converter that is connected to the PC. A separate USB-UART converter will be required per device to receive the terminal update in PC. The communication protocol of the device was available for one-to-one communication between the devices. Hence it was decided to implement the algorithm in two devices.

Sharing Spectrum Hole (Transaction) Data

When the device is switched on, it starts sensing and broadcasting its spectrum hole data to other devices in the network. Similarly, it also receives the spectrum data vector from other devices. Accordingly, the count of the received spectrum data is incremented. Transmission of the spectrum data is enabled by a timer interrupt that occurs two times in every 500ms. Since the transmission is carried out in Round-Robin method, the first device will transmit at 250ms and second device at 500ms. Hence the spectrum data is shared among devices every 500ms. The timer interrupt value will be divided according to the number of devices in the network. Since the device in this network is only 2, it was divided accordingly. In case of 3 devices, each device will transmit at a gap of approx 166ms in round robin method. The sensing timer can also be varied from 500ms to any value according to the requirement. The spectrum hole data (transaction) alongwith the device identity is stored in a data structure, which will be used to generate a block at the occurrence of PCA epoch. The screenshot at 4.6 will give the status update of a device as it receives the transaction from other devices.

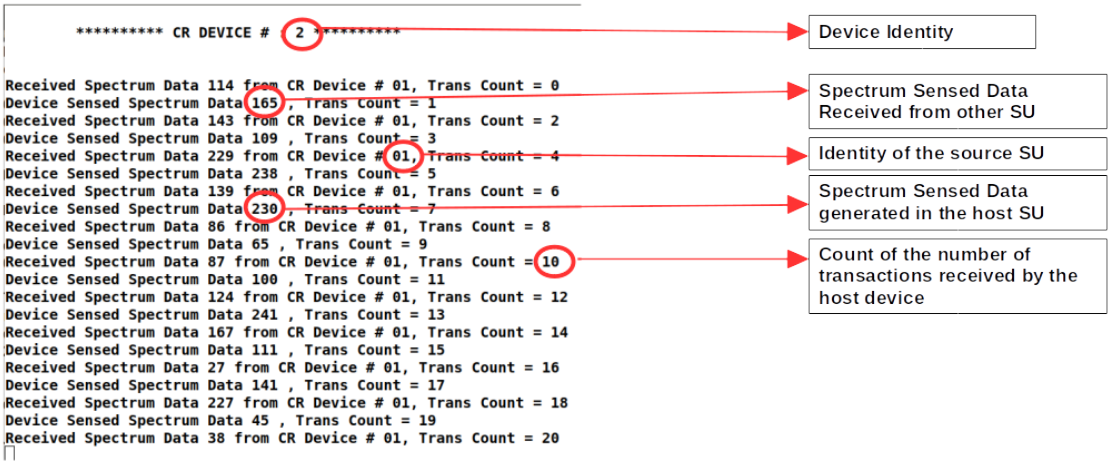

Figure 4.6: Device receiving Spectrum Data from other devices

Implementation of PCA Routine

The devices will keep sending and receiving the Spectrum Hole data in the network until the occurrence of PCA Epoch. PCA Epoch is based on a timer that is set to interrupt at every 10 seconds in this implementation. On the timer interrupt, the device with identity #1 will first transmit its transaction count to other devices, followed by other devices in ascending order of identity. Once all devices receives the transaction count data from other devices, then the winner is decided based on the reputation parameter calculation. The winner device then broadcast the block data to other devices and add the block to the chain. Other devices receive the block data and add it to their respective chain. On completion of these tasks, the winner device sends a synchronization message to other devices and restarts the timer for spectrum sensing and PCA. The details of the PCA Epoch are given in Fig. 4.7.

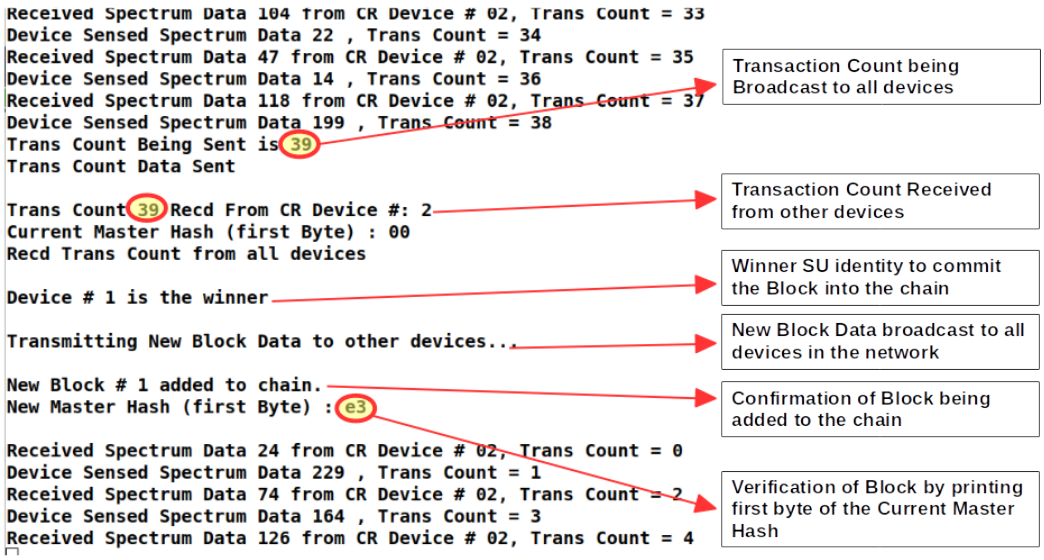

Figure 4.7: Devices carrying out PCA related tasks

The devices other than the winner will wait for the winner to transmit the block data. Upon receiving the new block data, they verify it and add the new block to the chain. Implementation of the same is given in Fig. 4.8

Implementation of PSC Routine

The epoch for carrying out ProBLeSS Smart Contract (PSC) will occur after every third PCA interrupt. In addition, verification of the received spectrum data will be car-

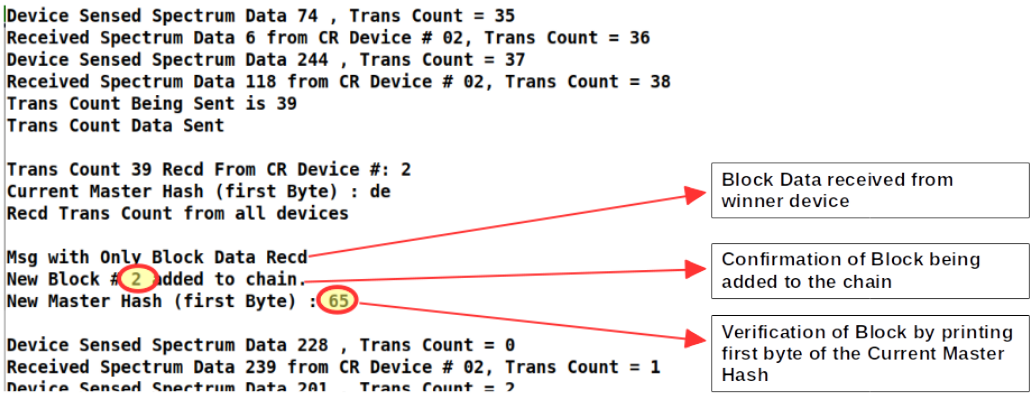

Figure 4.8: Devices carrying out PCA related tasks

ried out with the average channel availability of 10 latest committed blocks. The latest transactions received from a device will be verified and the result will be displayed. The verification is done by taking total count of all free and busy channel status of all transactions in the 10 latest blocks. The device does not support float point operations, hence the value was multiplied by 100 and divided with the total number of transactions. This is the average channel status in the committed blocks. The same availability is compared with the recently received transactions from other devices to confirm its validity. Since only two devices are in the network and the data is generated on random basis, the marking of a device as malicious after validating its data is not carried out in this implementation. The working of the same is given in Fig. 4.9

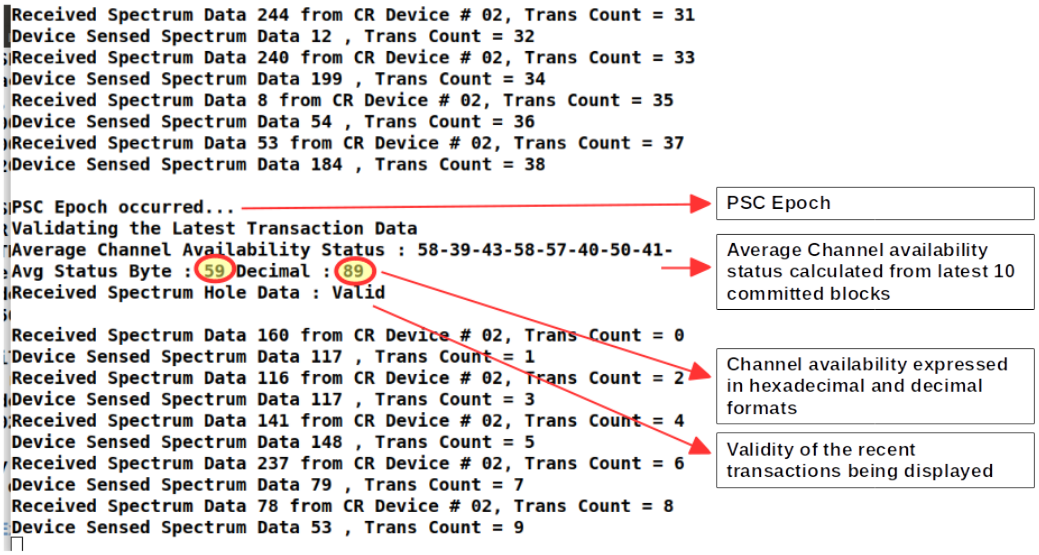

Figure 4.9: PSC for Validation of transactions

Memory Evaluation

The device memory is considered to be the most valuable resource in the devices like CR and IoTs. Hence the implementation of ProBLeSS was checked for the memory occupied by the elf format of the file using *size* command. The result of the same is given in Fig. The total memory available in Ubimote-SEZ is 32KB, the implementation used around 16KB memory. The blocks and transactions were stored in the main memory in this implementation. This could be further reduced if there is a flash memory in the CR device.

```
prabhu@GP:~/Git/Lab_Experiments/sdk/CR-MTP-1604/algo$ size algo.elf
   text    data     bss     dec     hex filename
  12992     705    2454   16151    3f17 algo.elf
prabhu@GP:~/Git/Lab_Experiments/sdk/CR-MTP-1604/algo$
```

Figure 4.10: Memory utilization

4.5 Simulation of ProBLeSS in CR Simulator

The actual performance of ProBLeSS algorithm in mitigating the SSDF attacks on a CR network can be understood only by simulating the same. The algorithm was simulated in an CR simulator and the results are discussed in this section.

4.5.1 Experimental Setup

It is assumed that there is a dedicated channel available for each PU in the network. The number of PUs, SUs and malicious SUs were varied to measure the parameters. It is also assumed that a CCC (Common Control Channel) exists between the SUs for transmission of control messages to share the blocks as well as the PCA parameters that are used to decide upon the *Winner*BE. The control messages are broadcast and identified by suitable headers. Every BE is loaded with a base hash H_b and maintains the H_m. Every SU that wants to transmit data uses the latest 7 blocks from the blockchain and uses the transaction data to make a spectrum decision using PROLEMus protocol [1]. In addition, every SU invokes the PCA and PSC once every 70s (PCA $epoch_C A$) based on the requirement of past 70 seconds spectrum data for best performance as suggested by the authors in [1]. The value of threshold is taken as 0.5 considering

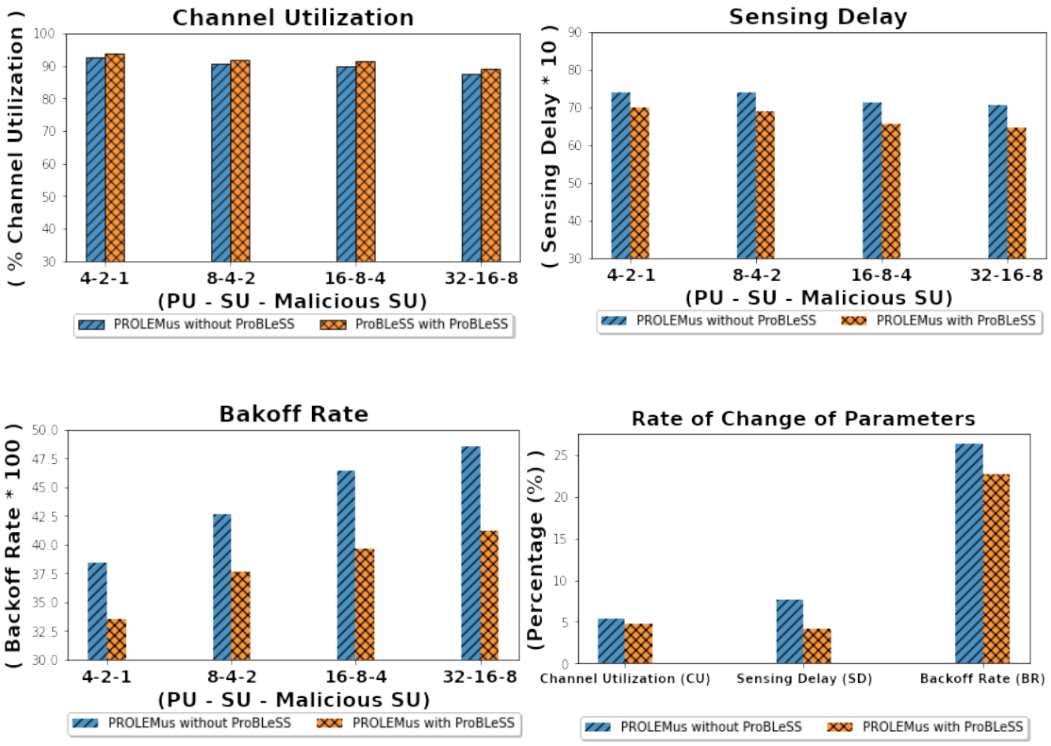

Figure 4.11: Performance comparison of PROLEMus with and without ProBLeSS Algorithm

a minimum *Reputation* of 0.5 for a worst case when *Backoff Rate* equals to the total number of channels.

The IoBT network was simulated for 24 hours with a malicious BE joining the network once every hour and transmitting false spectrum data for 10 to 30 minutes selected at random and also by selecting a SU as malicious (BE Capture) every hour to transmit false spectrum data. The average performance of the parameters for 24 hours have been taken for evaluation. The performance metrics used in the experiments are Channel Utilization (CU), Backoff Rate (BF) and Sensing Delay(SD). The CU is given as $[(t_{busy}) \div (t_{sim}) \times 100]$, where t_{sim} is total simulation time; and, t_{busy} is the total time when a channel was busy in transmission. The BR is calculated as [$N_{back} \div SU_{busy}$], where N_{back} is the total number of times the SUs backed off due to channels being busy; and, SU_{busy} is the total time when the SU was transmitting. The SD is calculated as [$N_{fail} \div SU_{pkts}$], where N_{fail} is the total number of channels sensed before any successful transmission; and, SU_{pkts} is the total number of SU packets transmitted.

The number of PUs, SUs and Malicious SUs in the network was changed from minimum to maximum possible numbers. Initially, the PUs were kept as 4 and gradually increased to 32, likewise the SUs were varied from 2 to 16. Similarly, the malicious

SUs were increased from 1 to 8. The malicious users were added into the network in the following ways to ascertain the performance of ProBLeSS:

(a) Experiment 1: The CR-IoBT network simulated with PROLEMus without ProBLeSS under SSDF attack by intelligent false spectrum data being generated by every malicious BE joining the network.

(b) Experiment 2: The CR-IoBT network simulated with PROLEMus using ProBLeSS under SSDF attack by false spectrum data being generated by every malicious BE joining the network.

The PROLEMus was run on the setup which uses its own Spectrum Sharing protocol and then exposed to SSDF attack. Thereafter, the PROLEMus was made to use ProBLeSS algorithm for spectrum sharing and exposed to SSDF attack. In both the configurations, the number of PUs, SUs and malicious users were varied from minimum to maximum and channels increased as per number of PUs.

The false spectrum data was generated by the malicious BE by using an intelligent SSDF which was simulated by marking that channel as busy where it was predicted by PROLEMus as free (FD_{intel}). The result of the experiments are shown in Fig 4.11.

4.5.2 Results

The following observations could be deducted from the experiment results.

(a) The rate at which channel utilization decreases with increase in malicious users in ProBLeSS is less than PROLEMus. However, the overall channel utilization increases marginally by 1.58% due to the blockchain overlay of ProBLeSS.

(b) The rate of sensing delay decreases with increase in number of SU and PU channels because of avail of more channels in ProBLeSS is more than PROLEMus. Also overall it decreases due to the blockchain overlay in ProBLeSS by avg 7.59%.

(c) The rate of Backoff Rate increases with the increase in malicious users in ProBLeSS is less than PROLEMus. However, there is a large decrease in the overall Backoff Rate due to the blockchain overlay in ProBLeSS by 13.54%.

CHAPTER 5

CONCLUSION

5.1 Conclusion

CR-IoBT networks are prone to SSDF attacks by malicious BEs who can influence the spectrum decision process by intelligently transmitting false spectrum data in a collaborative spectrum decision architecture. SSDF attacks can be prevented by BE authentication as well as security of spectrum data. However, traditional authentication mechanisms involve key management as well as cryptography protocols that make the resource constrained IoBT nodes inefficient. We propose a Proactive Blockchain based Spectrum Sharing protocol ProBLeSS that leverages the power of learning by proactive protocols like PROLEMus [1] for optimal performance and counters the SSDF attacks using blockchains without typical BE authentication. ProBLeSS was tested on a simulated network with an intelligent SSDF attack based on PROLEMus that has resulted in average 1.58 % increase in Channel Utilization, 7.59 % and 13.54 % reduction in Sensing Delay and Backoff Rate respectively when compared to PROLEMus under similar SSDF attack.

5.2 Future Work

The future work on the algorithm can be carried out on the following lines

(a) The algorithm was tested against the PROLEMus algorithm under SSDF attack with same setup. ProBLeSS can be evaluated against the other algorithms that are proposed to mitigate SSDF attacks. This will give an insight into the effectiveness of the ProBLeSS as against the similar algorithms.

(b) The blockchain infrastructure used against SSDF attacks can be further developed to mitigate other possible attacks in Cognitive Radios like Primary User Emulation Attack, Sinkhole Attack, control Channel Saturation DoS Attack, etc.

(c) Formally analyzing the algorithm to find optimal bounds for the parameters that are used to measure the performance.

(d) Implementation of the algorithm in a physical Cognitive Radio to check its functionality and its effect on the memory and computational resources of the radio.

Appendices

APPENDIX A

SETUP OF UBIMOTE-SEZ FOR PROGRAMMING

A.1 Hardware and Software Requirement

(a) Linux Ubuntu 16.04

(b) Segger Jlink-EDU Debugger

(c) USB to UART converter

A.2 Software Installation

(a) Update the standard repository of Ubuntu

sudo apt-get update

(b) Install the ARM cross compiler. The cross compiler is used to compile the code written in the PC for UbiMote

sudo apt-get install gcc-arm-none-eabi

(c) Install the serial terminal manager called picocom. Picocom is used to display as well as feed in the serial data

sudo apt-get install picocom

(d) Install the serial package for python in order to run the flasher scripts written in python

sudo apt-get install python-serial

(e) Copy the flasher script to 'bin' folder

sudo cp /Ubimote/Tools/Ubimote_Jlink_flasher /usr/bin

(f) Give executable permission for flasher script copied to bin folder

 # sudo chmod 0777 /usr/bin/Ubimote_Jlink_flasher

(g) Give executable permission for the flasher script which Is used when flashing the code via serial port.

 # sudo chmod 0777 /Ubimote/Lab_Experiments/SDK/ubimote-bsl

(h) Install the jlink debian package. This is the driver for jlink debugger.

 # sudo dpkg -i jlink_4.98.5_x86_64

A.3 Compiling and Flashing the Code

The programming can be done in two ways

(a) UART programming

(b) JTAG programming

(I) UART Programming

(a) The USB-UART converter has to be connected to the device and before programming, you have to make sure the device is in *bootloader* mode. You can put the device in *bootloader* mode by pressing the *reset* and *user* button together and release the *reset* switch and followed by *user* switch

(b) Goto the program path in the terminal and type "# make" and "# sudo make flash" commands

(II) JTAG Programming. When you are programming using the JTAG debugger, debugger should be connected to the 10 pin header and follow the steps

(a) Goto the program folder

(b) Open the terminal in that path and enter the commands "# make" and then "# sudo Ubimote_Jlink_flasher program *filename*.bin"

(III) Command to view the output on the terminal of the PC. The print statements in the program will send the data through serial interface to the USB-UART converter, 'picocom' terminal interface software can be used to see the output. The command for the same is given below.

(a) *sudo picocom -b 115200 /dev/ttyUSB0*

```
prabhu@GP:~/Git/Lab_Experiments/sdk/CR-MTP-1604/algo$ sudo picocom -b 115200 /dev/ttyUSB0
[sudo] password for prabhu:
picocom v2.2

port is        : /dev/ttyUSB0
flowcontrol    : none
baudrate is    : 115200
parity is      : none
databits are   : 8
stopbits are   : 1
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : no
nolock is      : no
send_cmd is    : sz -vv
receive_cmd is : rz -vv -E
imap is        :
omap is        :
emap is        : crcrlf,delbs,

Type [C-a] [C-h] to see available commands

Terminal ready
```

Figure A.1: Command Format : picocom

REFERENCES

[1] M. Patnaik, V. Kamakoti, Vashek Matyas, Vojtech Rehák, *"PROLEMus: A Proactive Learning-Based MAC Protocol Against PUEA and SSDF Attacks in Energy Constrained Cognitive Radio Networks"*, in IEEE Trans. Cogn. Comm. Networking, No. 5(2), pp. 400–412, 2019 .

[2] J. Mitola, *"Cognitive radio for flexible mobile multimedia communications"*, in Proc. IEEE International Workshop on Mobile Multimedia Communications (MoMuC), pp. 3–10, 1999.

[3] N. S. Narayanan, M. Patnaik, and V. Kamakoti, *"ProMAC: A proactive model predictive control based MAC protocol for cognitive radio vehicular networks"*, Computer Communications, Elsevier, vol. 93, pp. 27–38, 2016.

[4] Nakamoto Satoshi, *"Bitcoin: A Peer-to-Peer Electronic Cash System"*, in Cryptography Mailing list at https://metzdowd.com, 2009 .

[5] K. Kotobi and S. G. Bilén, *"Blockchain-enabled spectrum access in cognitive radio networks"*, in Wireless Telecommunications Symposium (WTS), Chicago, IL, 2017, pp. 1-6, 2017.

[6] S. Roy, M. Ashaduzzaman, M. Hassan, A. R. Chowdhury, *"BlockChain for IoT Security and Management: Current Prospects, Challenges and Future Directions"*, in 5th International Conference on Networking, Systems and Security, NSysS, 2019.

[7] Axel Moinet, Benoît Darties, Jean-Luc Baril, *"Blockchain based trust & authentication for decentralized sensor networks"*, arXiv:1706.01730v1 [cs.CR] 6 Jun 2017.

[8] D. Li, W. Peng, W. Deng and F. Gai, *"Blockchain based trust authentication for decentralized sensor networks"*, in 27th International Conference on Computer Communication and Networks, ICCCN , pp. 1095–2055, 2018.

[9] V. Buterin and V. Griffith, *"Casper the Friendly Finality Gadget"*, Ethereum Foundation, 2017.

[10] A. R. Sfar and E. Natalizio, Y. Challal and Z. Chtourou, *"A Roadmap for Security Challenges in Internet of Things"*, in Digital Communications and Networks, vol. 4, 2017.

[11] Merkle Tree "https://en.wikipedia.org/wiki/Merkle_tree" in Wikipedia

[12] Product page of CC2538 "https://www.ti.com/product/CC2538" TI website

LIST OF PAPERS BASED ON THESIS

(a) M. Patnaik, G. Prabhu, C. Rebeiro, V. Matyas and K. Veezhinathan, *"ProBLeSS: A Proactive Blockchain Based Spectrum Sharing Protocol Against SSDF Attacks in Cognitive Radio IoBT Networks,"* in IEEE Networking Letters, vol. 2, no. 2, pp. 67-70, June 2020, doi: 10.1109/LNET.2020.2976977.