International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 11, Issue 3 [May.-June. 2022] PP: 517-536

Attack-Proof Bidirectional Blockchain for energy efficient routing protocol for wireless sensor network

Dr.J S Kanchana
J Jenila
[1,2]*KLN college of Engineering, pottapalayam ,Madurai ,Tamilnadu.*

ABSTRACT:

The wireless sensor network (WSN) with fluctuating environments might be susceptible to various types of malicious attacks, and encryption algorithm to astound this challenge and mostly dependent on the authentication . Most predominant routing schemes in literature are fall backs in characterizing the malicious nodes on networks due to the real time variation of routing information. A bidirectional blockchain is fully based on authentication scheme and it is proposed for secure routing in the Wireless Sensor Network. The unauthentication and malicious node affect the routing process and to identify the correct routing path becomes a challenging issue. Bidirectional Blockchain is based on the multi-party computation and to ensure the secrecy of keys used to distributed data and integrity of the received data, respectively. Each and every node participates in the routing is authenticated by the base station .In the proposed routing protocol, a Cluster Head sends the data to BS by selecting the forwarder CH node based on the residual energy and minimum distance from BS . The simulation results show that our proposed model improves the packet delivery ratio and the network lifetime is also increased**.**

KEYWORDS:
Authentication,Blockchain,Bidirectional,WSN (wireless sensor network),BS(Base Station)

Date of Submission: 07-06-2022 Date of Acceptance: 22-06-2022

---

1. INTRODUCTION

A network can be characterized as wired or wireless. Wireless can be distinguished from wired as no physical connectivity between nodes are needed. Routing is an activity or a function that connects a call from origin to destination in telecommunication networks and also play an important role in architecture, design and operation of networks. It deals with more and more details related to routing and its concepts.
Ad-hoc networks are wireless networks where nodes communicate with each other using multi-hop links. There is no stationary infrastructure or base station for communication. Each node itself acts as a router for forwarding and receiving packets to/from other nodes. Routing in ad-networks has been a challenging task ever since the wireless networks came into existence. The major reason for this is the constant change in network topology because of high degree of node mobility. A number of protocols have been developed for accomplish this task. Some of them are DSDV and AODV routing protocols which are explained in the forthcoming chapters.

1.1 ROUTING

Routing is the act of moving information from a source to a destination in an internetwork. During this process, at least one intermediate node within the internetwork is encountered. This concept is not new to computer science since routing was used in the networks in early 1970's. But this concept has achieved popularity from the mid-1980's. The major reason for this is because the earlier networks were very simple and homogeneous environments; but, now high end and large scale internetworking has become popular with the latest advancements in the networks and telecommunication technology.
The routing concept basically involves, two activities: firstly, determining optimal routing paths and secondly, transferring the information groups (called packets) through an internetwork. The later concept is called as packet switching which is straight forward, and the path determination could be very complex. Routing protocols use several metrics to calculate the best path for routing the packets to its destination. These metrics

are a standard measurement that could be number of hops, which is used by the routing algorithm to determine the optimal path for the packet to its destination. The process of path determination is that, routing algorithms initialize and maintain routing tables, which contain the total route information for the packet. This route information varies from one routing algorithm to another.
Routing tables are filled with a variety of information which is generated by the routing algorithms. Most common entries in the routing table are ip-address prefix and the next hop. Routing table's Destination/next hop associations tell the router that a particular destination can be reached optimally by sending the packet to a router representing the "next hop" on its way to the final destination and ip-address prefix specifies a set of destinations for which the routing entry is valid for.
Switching is relatively simple compared with the path determination. The concept of switching is like, a host determines like it should send some packet to another host. By some means it acquires the routers address and sends the packet addressed specifically to the routers MAC address, with the protocol address of the destination host. The router then examines the protocol address and verifies whether it know how to transfer the data to its destination. If it knows how to transfer the data then it forwards the packet to its destination and if it doesn't then it drops the packet.
Routing is mainly classified into static routing and dynamic routing. Static routing refers to the routing strategy being stated manually or statically, in the router. Static routing maintains a routing table usually written by a networks administrator. The routing table doesn't depend on the state of the network status, i.e., whether the destination is active or not. Dynamic routing refers to the routing strategy that is being learnt by an interior or exterior routing protocol. This routing mainly depends on the state of the network i.e., the routing table is affected by the activeness of the destination.
The major disadvantage with static routing is that if a new router is added or removed in the network then it is the responsibility of the administrator to make the necessary changes in the routing tables. But this is not the case with dynamic routing as each router announces its presence by flooding the information packet in the network so that every router within the network learn about the newly added or removed router and its entries. Similarly this is the same with the network segments in the dynamic routing.

1.2 Classification of Dynamic Routing Protocols

Dynamic routing protocols are classified depending on what the routers tell each other and how they use the information to form their routing tables. They are Distance vector protocols and Link state protocols Most of the protocols available in the networks fit into one of the two categories.

1.2.1 Distance Vector Protocols

By using the distance vector protocols, each router over the internetwork send the neighboring routers, the information about destination that it knows how to reach. Moreover to say the routers sends two pieces of information first, the router tells, how far it thinks the destination is and secondly, it tells in what direction (vector) to use to get to the destination. When the router receives the information from the others, it could then develop a table of destination addresses, distances and associated neighboring routers, and from this table then select the shortest route to the destination. Using a distance vector protocol, the router simply forwards the packet to the neighboring host (or destination) with the available shortest path in the routing table and assumes that the receiving router will know how to forward the packet beyond that point. The best example for this is the routing information protocol (RIP).

1.2.2 Link-State Protocols

In link state protocols, a router doesn't provide the information about the destination instead it provides the information about the topology of the network. This usually consist of the network segments and links that are attached to that particular router along with the state of the link i.e., whether the link is in active state or the inactive state. This information is flooded throughout the network and then every router in the network then builds its own picture of the current state of all the links in the network.

1.3 Mobile Ad-hoc Networks

An ad-hoc network is a collection of wireless mobile hosts forming a temporary network without the aid of any stand-alone infrastructure or centralized administration. Mobile Ad-hoc networks are self-organizing and self configuring multihop wireless networks where, the structure of the network changes dynamically. This is mainly due to the mobility of the nodes. Nodes in these networks utilize the same random access wireless channel,

cooperating in a friendly manner to engaging themselves in multihop forwarding. The nodes in the network not only acts as hosts but also as routers that route data to/from other nodes in network.
In mobile ad-hoc networks where there is no infrastructure support as is the case with wireless networks, and since a destination node might be out of range of a source node transmitting packets; a routing procedure is always needed to find a path so as to forward the packets appropriately between the source and the destination. Within a cell, a base station can reach all mobile nodes without routing via broadcast in common wireless networks. In the case of ad-hoc networks, each node must be able to forward data for other nodes. This creates additional problems along with the problems of dynamic topology which is unpredictable connectivity changes.

1.3.1 Problems with routing in Mobile Ad-hoc Networks

Asymmetric links: Most of the wired networks rely on the symmetric links which are always fixed. But this is not a case with ad-hoc networks as the nodes are mobile and constantly changing their position within network. For example consider a MANET(Mobile Ad-hoc Network) where node B sends a signal to node A but this does not tell anything about the quality of the connection in the reverse direction.
Routing Overhead: In wireless adhoc networks, nodes often change their location within network. So, some stale routes are generated in the routing table which leads to unnecessary routing overhead.
Interference: This is the major problem with mobile ad-hoc networks as links come and go depending on the transmission characteristics, one transmission might interfere with another one and node might overhear transmissions of other nodes and can corrupt the total transmission.
Dynamic Topology: This is also the major problem with ad-hoc routing since the topology is not constant. The mobile node might move or medium characteristics might change. In ad-hoc networks, routing tables must somehow reflect these changes in topology and routing algorithms have to be adapted. For example in a fixed network routing table updating takes place for every 30sec. This updating frequency might be very low for ad-hoc networks.

1.4 Classification of routing Protocols in MANET’s

Classification of routing protocols in MANET’s can be done in many ways, but most of these are done depending on routing strategy and network structure. According to the routing strategy the routing protocols can be categorized as Table-driven and source initiated, while depending on the network structure these are classified as flat routing, hierarchical routing and geographic position assisted routing. Both the Table-driven and source initiated protocols come under the Flat routing.

1.4.1 Table-Driven routing protocols

These protocols are also called as proactive protocols since they maintain the routing information even before it is needed. Each and every node in the network maintains routing information to every other node in the network. Routes information is generally kept in the routing tables and is periodically updated as the network topology changes. Many of these routing protocols come from the link-state routing. There exist some differences between the protocols that come under this category depending on the routing information being updated in each routing table. Furthermore, these routing protocols maintain different number of tables. The proactive protocols are not suitable for larger networks, as they need to maintain node entries for each and every node in the routing table of every node. This causes more overhead in the routing table leading to consumption of more bandwidth.

1.4.2 On Demand routing protocols

These protocols are also called reactive protocols since they don’t maintain routing information or routing activity at the network nodes if there is no communication. If a node wants to send a packet to another node then this protocol searches for the route in an on-demand manner and establishes the connection in order to transmit and receive the packet. The route discovery usually occurs by flooding the route request packets throughout the network.

1.4.3 Destination Sequenced Distance Vector (DSDV) Protocol

The destination sequenced distance vector routing protocol is a proactive routing protocol which is a modification of conventional Bellman-Ford routing algorithm. This protocol adds a new attribute, sequence number, to each route table entry at each node. Routing table is maintained at each node and with this table, node transmits the packets to other nodes in the network. This protocol was motivated for the use of data exchange along changing and arbitrary paths of interconnection which may not be close to any base station.

1.4.4 Protocol Overview and activities

Each node in the network maintains routing table for the transmission of the packets and also for the connectivity to different stations in the network. These stations list for all the available destinations, and the number of hops required to reach each destination in the routing table. The routing entry is tagged with a sequence number which is originated by the destination station. In order to maintain the consistency, each station transmits and updates its routing table periodically. The packets being broadcasted between stations indicate which stations are accessible and how many hops are required to reach that particular station. The packets may be transmitted containing the layer 2 or layer 3 address.
Routing information is advertised by broadcasting or multicasting the packets which are transmitted periodically as when the nodes move within the network. The DSDV protocol requires that each mobile station in the network must constantly, advertise to each of its neighbors, its own routing table. Since, the entries in the table my change very quickly, the advertisement should be made frequently to ensure that every node can locate its neighbors in the network. This agreement is placed, to ensure the shortest number of hops for a route to a destination; in this way the node can exchange its data even if there is no direct communication link. The data broadcast by each node will contain its new sequence number and the following information for each new route:
The destination address
The number of hops required to reach the destination and
The new sequence number, originally stamped by the destination
The transmitted routing tables will also contain the hardware address, network address of the mobile host transmitting them. The routing tables will contain the sequence number created by the transmitter and hence the most new destination sequence number is preferred as the basis for making forwarding decisions. This new sequence number is also updated to all the hosts in the network which may decide on how to maintain the routing entry for that originating mobile host.
After receiving the route information, receiving node increments the metric and transmits information by broadcasting. Incrementing metric is done before transmission because, incoming packet will have to travel one more hop to reach its destination.
Time between broadcasting the routing information packets is the other important factor to be considered. When the new information is received by the mobile host it will be retransmitted soon effecting the most rapid possible dissemination of routing information among all the cooperating mobile hosts. The mobile host cause broken links as they move form place to place within the network. The broken link may be detected by the layer2 protocol, which may be described as infinity. When the route is broken in a network, then immediately that metric is assigned an infinity metric there by determining that there is no hop and the sequence number is updated. Sequence numbers originating from the mobile hosts are defined to be even number and the sequence numbers generated to indicate infinity metrics are odd numbers.
The broadcasting of the information in the DSDV protocol is of two types namely: full dump and incremental dump. Full dump broadcasting will carry all the routing information while the incremental dump will carry only information that has changed since last full dump. Irrespective of the two types, broadcasting is done in network protocol data units (NPDU). Full dump requires multiple NPDU's while incremental requires only one NPDU to fit in all the information.
When an information packet is received from another node, it compares the sequence number with the available sequence number for that entry. If the sequence number is larger, then it will update the routing information with the new sequence number else if the information arrives with the same sequence number it looks for the metric entry and if the number of hops is less than the previous entry the new information is updated (if information is same or metric is more then it will discard the information). While the nodes information is being updated the metric is increased by 1 and the sequence number is also increased by 2. Similarly, if a new node enters the network, it will announce itself in the network and the nodes in the network update their routing information with a new entry for the new node.
During broadcasting, the mobile hosts will transmit their routing tables periodically but due to the frequent movements by the hosts in the networks, this will lead to
continuous burst of new routes transmissions upon every new sequence number from that destination. The solution for this is to delay the advertisement of such routes until it shows up a better metric.

Advantages of DSDV

DSDV protocol guarantees loop free paths.
Count to infinity problem is reduced in DSDV.
We can avoid extra traffic with incremental updates instead of full dump updates.
Path Selection: DSDV maintains only the best path instead of maintaining multiple paths to every destination. With this, the amount of space in routing table is reduced.

Limitations of DSDV

Wastage of bandwidth due to unnecessary advertising of routing information even if there is no change in the network topology.
DSDV doesn't support Multi path Routing.
It is difficult to determine a time delay for the advertisement of routes
It is difficult to maintain the routing table's advertisement for larger network. Each and every host in the network should maintain a routing table for advertising. But for larger network this would lead to overhead, which consumes more bandwidth.

1.4.5 Ad-hoc On-Demand Distance Vector (AODV) Protocol

AODV is a very simple, efficient, and effective routing protocol for Mobile Ad hoc Networks which do not have fixed topology. This algorithm was motivated by the limited bandwidth that is available in the media that are used for wireless communications.
It borrows most of the advantageous concepts from DSR and DSDV algorithms. The on demand route discovery and route maintenance from DSR and hop-by-hop routing, usage of node sequence numbers from DSDV make the algorithm cope up with topology and routing information. Obtaining the routes purely on-demand makes AODV a very useful and desired algorithm for MANETs.

Working of AODV

Each mobile host in the network acts as a specialized router and routes are obtained as needed, thus making the network self-starting. Each node in the network maintains a routing table with the routing information entries to it's neighbouring nodes, and two separate counters: a node sequence number and a broadcast-id. When a node (say, source node 'S') has to communicate with another (say, destination node 'D'), it increments its broadcast-id and initiates path discovery by broadcasting a route request packet RREQ to its neighbors. The RREQ contains the following fields: – source-addr
– source-sequence# - to maintain freshness info about the route to the source.
– dest-addr
– dest-sequence# - specifies how fresh a route to the destination must be before it is accepted by the source.
– hop-cnt
The (source-addr, broadcase-id) pair is used to identify the RREQ uniquely. Then the dynamic route table entry establishment begins at all the nodes in the network that are on the path from S to D.
As RREQ travels from node to node, it automatically sets up the reverse path from all these nodes back to the source. Each node that receives this packet records the address of the node from which it was received. This is called Reverse Path Setup. The nodes maintain this info for enough time for the RREQ to traverse the network and produce a reply to the sender and time depends on network size.
If an intermediate node has a route entry for the desired destination in its routing table, it compares the destination sequence number in its routing table with that in the RREQ. If the destination sequence number in its routing table is less than that in the RREQ, it rebroadcasts the RREQ to its neighbors. Otherwise, it unicasts a route reply packet to its neighbor from which it was received the RREQ if the same request was not processed previously (this is identified using the broadcase-id and source-addr).
Once the RREP is generated, it travels back to the source, based on the reverse path that it has set in it until traveled to this node. As the RREP travels back to source, each node along this path sets a forward pointer to the node from where it is receiving the RREP and records the latest destination sequence number to the request destination. This is called Forward Path Setup. If an intermediate node receives another RREP after propagating the first RREP towards source it checks for destination sequence number of new RREP. The intermediate node updates routing information and propagates new RREP only,
– If the Destination sequence number is greater, OR
– If the new sequence number is same and hop count is small, OR

Otherwise, it just skips the new RREP. This ensures that algorithm is loop-free and only the most effective route is used.

1.5 Attacks in WSN

The WSNs are more susceptible to security attacks rather than to wired networks. Due to the facts such as restricted protection of every individual node, uneven behaviour of connectivity, deficit of certification authority, centralized monitoring or administration, security is complicated aspect to be maintained in these networks. In such a wireless network, attacks can enter from all possible directions and focus at any node. Hence, each node is priorly ready for facing attacks straightly or in an indirect manner. In particular, an attack from a compromised node inside the network is destructive and difficult to be identified. MANETs are exposed to both passive and active attacks. During active attacks, the adversary does replication, alteration and removal of swapped data, while in passive attacks, it results in eavesdropping of data. In particular, the attacks in such a network can result in congestion, spreading wrong routing information, avoiding regular functioning of services or complete shutdown.
The weaknesses of ad hoc networks are dynamic topology, lack of infrastructure, exposure of nodes and channels.

1.5.1 Intrusion detection system (IDS)

As the initial step to prevent the active attack, the technique such as authentication and encryption can be taken into account. Though these techniques have certain issues, they are specially designed for particular set of identified attacks. They are not suitable for new attack. Hence, a technique called 'Intrusion detection system' is defined for identifying and reacting to newer attacks.
An IDS is a method for recognising a group of events, which tries to negotiate the integrity, privacy or ease of use of resources. This method is related to scheme that tries to identify intrusion into a computer or network by witnessing certain events, security logs or audit data. In particular, IDS supervises the behaviour of the system and investigates the activities in order to verify whether any event is disrupting the security rules. In case there is occurrence of strange activity or any attack, IDS generates an alarm to notify the security administrator. Also, IDS can generate an appropriate reply to the malicious activity.
There are three categories of IDS which are designed to be suitable for MANET and are described below.
Stand-alone IDS: In this architecture, every individual node has separate IDS for detecting intrusions. There is no mutual aid or interchange of data among themselves in these IDSs. This design is more appropriate for flat network rather than for multilayered network communication.
Distributed and cooperative (DAC) IDS: In this architecture, every individual node in the MANET takes part in the detection of intrusion. This system reacts via IDS agent functioning over them. The IDS agent identifies and gathers local activities and data for categorising the feasible intrusion and also reacts separately.
Distributed and dictatorial (DAD) or DAC distributes audit data collection components via the network while the detection of intrusion is performed imperiously by a centralised host in the DAD architecture and considerately performed by contributing hosts in the DAC architecture.
Hierarchical IDS: The hierarchical IDS is an extension of DAC architecture. This is mainly projected for a multilayered network infrastructure. The network is partitioned into clusters. Each cluster is headed by the cluster head. The IDS agents function on each individual member node that monitors and takes decisions regarding identified intrusions. The cluster head is in charge of monitoring the network packets and commences the global action during the detection of intrusion.

1.5.2 Issues of IDS in WSN

The inbuilt difficulties in WSN cause the intrusion detection a complicated methodology and result in complexities in transmitting the wired intrusion detection method to the MANET atmosphere.
. Shortage of centralised management.
. Restricted accessibility of bandwidth.
. Dynamic connectivity.
. Utilisation of end-to-end cryptography .
In addition to the above phenomenon, other issues related to IDS offered by the MANETs are as follows:
. The restricted feature in IDS on WSN is to investigate the traffic within the range of wireless radio.
. The mobility of the nodes.
. In WSN, the nodes are more susceptible to compromise attacks.

. Owing to the dynamic topology of the network, an IDS is not capable to acquire sufficient data for perfect detection of the intrusion.
. Due to the deficit of flexibility, the methodology cannot assure approval of varied performance and cost necessities such as consumption of bandwidth, sensing precision, unremitting accessibility and latency detection.
. The existing IDS is refrained particularly for spotting recognised services level network attacks. The network manager is permitted to sense the policy intrusions from the currently available data or gathered data. As there are huge data, the investigation of data involves more wastage of time.
. Due to the existence of the several false positives and restricted resources for investigating the data for policy intrusions, there is more expenditure of the restricted resources.

2. PROJECT DESCRIPTION

2.1 Reactive Routing

Reactive Protocols use a route discovery process to flood the network with route query requests when a packet needs to be routed using source routing or distance vector routing. Source routing uses data packet headers containing routing information meaning nodes don’t need routing tables; however this has highnetwork overhead. Distance vector routing uses next hop and destination addresses to route packets, this requires nodes to store active routes information until no longer required or an active route timeout occurs, this prevents stale routes . Flooding is a reliable method of disseminating information over the network, however it uses bandwidth and creates network overhead, reactive routing broadcasts routing requests whenever a packet needs routing, this can cause delays in packet transmission as routes are calculated, but features very little control traffic overhead and has typically lower memory usage than proactive alternatives, this increases the scalability of the protocol

2.2 Hybrid Routing

Hybrid protocols combine features from both reactive and proactive routing protocols, typically attempting to exploit the reduced control traffic overhead from proactive systems whilst reducing the route discovery delays of reactive systems by maintaining some form of routing table . Survey papers successfully collect information from a wide range of literature and provide detailed and extensive reference material for attempting to deploy a WSN, both papers reach the conclusion that no single WSN routing protocol is best for every situation meaning analysis of the network and environmental requirements is essential for selecting an effective protocol. Whilst these papers contain functionality details for many of the protocols available, performance information for the different protocols is very limited and no details of any testing methodologies is provided, because of this the validity of some claims made cannot be verified.

2.3 EARLY WSN ROUTING PROTOCOLS

The next piece of literature is a protocol performance comparison by which compares the proactive Destination Sequenced Distance Vector (DSDV) protocol and the reactive Dynamic Source Routing (DSR) protocol; these protocols were developed in 1994 and were amongst the earliest MANET routing protocols identified using the previous survey papers.

A. Destination Sequenced Distance Vector (DSDV)

The proactive DSDV protocol was proposed by and is based upon the Bellman-Ford algorithm to calculate the shortest number of hops to the destination . Each DSDV node maintains a routing table which stores; destinations, next hop addresses and number of hops as well as sequence numbers; routing table updates are sent periodically as incremental dumps limited to a size of 1 packet containing only new information . DSDV compensates for mobility using sequence numbers and routing table updates, if a route update with a higher sequence number is received it will replace the existing route thereby reducing the chance of routing loops, when a major topology change is detected a full routing table dump will be performed, this can add significant overhead to the network in dynamic scenarios

B. Dynamic Source Routing (DSR)

The reactive DSR Protocol is broken into two stages; route discovery phase and route maintenance phase, these phases are triggered on demand when a packet needs routing. Route discovery phase floods the network with route requests if a suitable route is not available in the route . DSR uses a source routing strategy to generate a complete route to the destination, this will then be stored temporarily in nodes route cache . DSR addresses mobility issues through the use of packet acknowledgements; failure to receive an acknowledgement causes packets to be be buffered and route error messages to be sent to all upstream nodes. Route error messages trigger the route maintenance phase which removes incorrect routes from the route cache and undertakes a new route discovery phase

C. Mobility Models

Compares the performance of DSR and DSDV using simulations against 4 different mobility models; these are mathematic models which control the motion of nodes around the simulation; this allows researchers to measure the effect of mobility upon the routing protocols performance. Various mobility models are used to simulate different situations such as high speed vehicular networks or lower mobility ad-hoc conference users, however reveals that many studies perform protocol evaluation almost exclusively using the random waypoint mobility model. This research is supported by findings who claim that the random waypoint model is the most widely used mobility model, however discrepancies were identified between the models behaviour and real world scenarios where users typically move in groups, due to this the model may not be appropriate for exclusive testing.

The comparison among the routing protocols
In general, routing protocols for the wireless sensor networks can be divided into flat-based, hierarchical-based, and location-based routing protocol on the basis of the underlying network structure .
Flat-based routing protocol
Sensor nodes in flat-based routing protocols have the same role and collaborate together to perform the sensing task and perform multi-hop communication. Since the flat routing is based on flooding, it has several demerits, such as routing overhead and highly energy consumption.
Hierarchical-based routing protocol
In hierarchical-based routing protocols, the network is divided into several logical groups by a fixed area. The logical groups are called a cluster. Sensor nodes collect the information in the cluster and a head node aggregates the information to decrease the amount of the data. Each sensor node delivers the sensing data to a head node in the cluster and the head node delivers an aggregated data to the base station which is located outside of the sensor networks.
In general, a node which has more energy than neighbor sensor nodes is selected as a head node. Contrary to flat routing protocols, only a head node aggregates the collected information and sends it to the base station. Due to these advantages, sensor nodes can save their own energy remarkable.
Location-based routing protocol
This protocol is based on the location information of senor nodes in the wireless sensor networks. It assumes that each node would know its own location and a neighbor sensor node's location before sensor nodes sensing and collect the peripheral information. The distance between neighboring sensor nodes can be computed on the basis of the incoming signal strength.

2.4 BLOCKCHAIN

The term “blockchain technology” before, in reference to Bitcoin and other crypto currencies, for the uninitiated, the term might seem abstract with little real meaning on the surface. However, blockchain technology is a critical element of crypto currencies — without it, digital currencies like Bitcoin would not exist. If you are new to crypto currencies, and new to block chain technology, read this guide on the basics to get yourself started. If you are already a seasoned trader, maybe you’ll learn a thing or two you didn’t already know.

Blockchain technology
The idea of decentralization. By design, the **block chain** is a decentralized **technology**. A global network of computers uses **block chain technology** to jointly manage the database that records Bitcoin transactions. That is, Bitcoin is managed by its network, and not any one central authority.

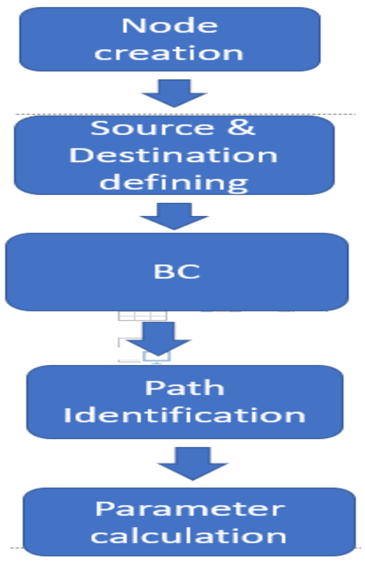
Working of Blockchain:

So, then, how does the blockchain work? Let's recall a few key features before we get into the details:

1. Blockchain keeps a record of all data exchanges — this record is referred to as a "ledger" in the cryptocurrency world, and each data exchange is a "transaction". Every verified transaction
2. It utilizes a distributed system to verify each transaction — a peer-to-peer network of nodes
3. Once signed and verified, the new transaction is added to the blockchain and can not be altered

To begin, we need to explore the concept of "keys". With a set of cryptographic keys, you get a unique identity. Your keys are the Private Key and Public Key, and together they are combined to give you a digital signature. Your public key is how others are able to identify you. Your private key gives you the power to digitally sign and authorize different actions on behalf of this digital identity when used with your public key.

In the proposed routing mechanism, the CH communicates directly with the BS if it is located in the transmission range. Otherwise, the node selects one of its neighbor nodes for packet forwarding based on their residual energy and distance from the BS. When a node A wants to communicate with another node B, it sends a communication request to the blockchain via smart contract. In response, the BS checks the reputation of B stored in the blockchain. If the reputation of B is higher than the predefined threshold value, a confirmation message is sent to A. The same procedure is followed by B to verify the identity of A, Thus a mutual authentication is performed and A sends the data packets to B. After continuous transmission, the data packets reach the BS. After receiving the data, the BS requests CH to authenticate the node.In response, the CH checks if it has the information of the node. After verification, CH sends the acknowledgment message to the BS. At this stage, if node is not registered then CH provides a negative feedback to BS and node is declared as a malicious node. Consequently, the reputation of the node is decreased. On the other hand, if the verification of the node is successful, its reputation is increased.



Blockchain technology is only a decade old. This means that it is a new technology that requires time to mature. If you take the different consortium into account, you will notice multiple players trying to solve the decentralized problem with their unique solution.

blockchain technology — which is not feasible for every business out there.

SOURCE CODE:

```
import math
import random
import sys
import pylab
import time
import matplotlib.pyplot as plt
import numpy as np
def distance(p1,p2):
   return math.sqrt( (p1[0]-p2[0])**2 + (p1[1]-p2[1])**2 )
class MobileModel:
   maxWidth = 1000
   maxHeight = 1000
   scale = 10
```

```
  minSpeed = 0
  maxSpeed = 150
  time = 1
  def __init__(self):
    pass
    def randomSpeed(self):
    return random.randint(self.minSpeed,self.maxSpeed)
  def randomWidth(self):
    return
random.randint(-self.maxWidth/self.scale,self.maxWidth/self.scale)
  def randomHeight(self):
    return random.randint(-self.maxHeight/self.scale,self.maxHeight/self.scale)
  def move(self):
    pass
class Assigning_random(MobileModel):
  currentnode_Pos = [0,0]
  next_Pos = [0,0]
  currentnode_Speed = 0;
  def __init__(self, currentnode_Pos, next_Pos, currentnode_Speed, *arg):
    self.currentnode_Pos = currentnode_Pos
    self.next_Pos = next_Pos
    self.currentnode_Speed = currentnode_Speed
    self.maxWidth ,self.maxHeight, self.scale ,self.minSpeed, self.maxSpeed, self.time = arg
  def move(self):
    if self.currentnode_Speed == 0:
      self.setSpeed(self.randomSpeed())
      retu
    if( distance( self.currentnode_Pos,self.next_Pos ) < self.currentnode_Speed ):
      self.setcurrentnode_Pos(self.next_Pos)
      self.setSpeed(self.randomSpeed())
      self.setnext_Pos(self.nextWayPoint())
    else:         self.setcurrentnode_Pos(self.nextcurrentnode_Point())
  def setcurrentnode_Pos(self,currentnode_Pos):
    self.currentnode_Pos = currentnode_Pos
  def setnext_Pos(self,next_Pos):
    self.next_Pos = next_Pos
  def setSpeed(self,currentnode_Speed):
    self.currentnode_Speed = currentnode_Speed
    def divideSpeed(self):
    x = abs(self.currentnode_Pos[0] - self.next_Pos[0])
```

```
        y = abs(self.currentnode_Pos[1] - self.next_Pos[1])
        if x==0 and y==0:
            return 0, 0
                l = distance([x,y], [0,0])
        return self.currentnode_Speed * x / l, self.currentnode_Speed * y/ l
    def nextWayPoint(self):
      x = self.currentnode_Pos[0]+self.randomWidth()
     y = self.currentnode_Pos[1]+self.randomHeight()
        while x < 0 or y < 0 or x > self.maxWidth or y > self.maxHeight:
     x = self.currentnode_Pos[0]+self.randomWidth()
     y = self.currentnode_Pos[1]+self.randomHeight()
        return x, y
        def nextcurrentnode_Point(self):
        vx, vy = self.divideSpeed()
        x, y = self.currentnode_Pos
        if( self.next_Pos[0] >= self.currentnode_Pos[0] ):
            x = x + vx * self.time
        else:
            x = x - vx * self.time
        if( self.next_Pos[1] >= self.currentnode_Pos[1] ):
            y = y + vy * self.time
        else:
            y = y - vy * self.time
        return x, y
class RouteProtocl:
    pcktQ = {}
    stsQ = {}
    routeTable = {}
    def __init__(self):
        pass
    def Sender(self):
        pass
    def Receiver(self):
        pass
class HierClust(RouteProtocl):
    addr = 0
    broadcast_id = 0
    sequenceNo = 0;
       Received_Req_Q = []
    dataQueue = []
    def __init__(self, addr):
        self.addr = addr
        self.pcktQ = dict( zip( ('rcv', 'snd'), ([],[]) ) )
        self.stsQ = dict( zip( ('rcvd','sndd'),       (dict(zip(('HELLO','RREQ','RREP','RERR','DATA'),([],[],[],[],[])))
,dict(zip(('HELLO','RREQ','RREP','REER','DATA'),
                                          ([],[],[],[],[]) ) ) )))
        self.Received_Req_Q = list()
        self.dataQueue = list()
        self.routeTable = dict()
        self.pcktQ['snd'].append([self.addr, 'broadcast', ['HELLO', self.addr,100] ])
        self.pcktQ['snd'].append([self.addr, 'broadcast', ['RREQ', 11, 99, 11, 20, 99, 5] ] )
    def timer(self):
        rec = []
        for index, entry in zip(range(sys.maxsize), self.Received_Req_Q):
            entry[0] -= 1
            if entry[0] == 0:
                print ('RREQ TIME out!')
                rec.append(index)
        rec.reverse()
        for index in rec:
            self.Received_Req_Q.pop(index)
        rec = []
        for key in self.routeTable:
            self.routeTable[key]['expirTime'] -= 1
            if self.routeTable[key]['expirTime'] == 0:
                print ('RouteTable timeout!')
                rec.append(key)
        for key in rec:
            self.routeTable.pop(key)
    def pcktGeneration(self, pckt):
        self.pcktQ['snd'].append(pckt)
    def Data_pckt_Creation(self, destAddr, numOfPakcet):
        for i in range(numOfPakcet):
            self.dataQueue.append(['DATA', self.addr, destAddr, i])
        def Data_pckt_Formation(self):
        if len(self.dataQueue) == 0:
            return
        data = self.dataQueue[0]
        if self.routeTable.__contains__(data[2]):
            self.pcktQ['snd'].append( [self.addr, self.routeTable[data[2]]['ntHop'], data] )
            self.dataQueue.pop(0)
        else:
            pckt =  [self.addr, 'broadcast', ['RREQ', self.addr, self.sequenceNo, self.broadcast_id, data[2], 0, 0]]
            for t, entry in self.Received_Req_Q:
                if entry[1] == pckt[2][1]:
                    return False
            self.pcktQ['snd'].append( pckt )
            self.broadcast_id += 1;
    def Sender(self):
        self.Data_pckt_Formation()
        pckt = self.Send_packt_process()
        return pckt
    def Receiver(self, pckt):
        if  pckt != False and pckt[0] != self.addr:
            self.pcktQ['rcv'].append(pckt)
        self.BlockChainRecev()
    def Send_Hello_pckt(self, pckt):
        pass
    def Send_RREQ_pckt(self, pckt):
        self.Received_Req_Q.append( [4, pckt[2]] )

    def Send_RREP_pckt(self, pckt):
        time.sleep(random.randint(1,3))
        pass
    def Send_RERR_pckt(self, pckt):
        pass
    def Send_Data_pckt(self, pckt):
        pass
    def Send_packt_process(self):
        sendpckt_dispatch          =          {'HELLO':self.Send_Hello_pckt,          'RREQ':self.Send_RREQ_pckt,
'RREP':self.Send_RREP_pckt,
                        'RERR':self.Send_RERR_pckt, 'DATA':self.Send_Data_pckt}
        try:
            pckt = self.pcktQ['snd'].pop(0)
            sendpckt_dispatch[pckt[2][0]](pckt)
            self.stsQ['sndd'][pckt[2][0]].append(pckt[2])
            return pckt
        except:
            return False
    def helloPktReceived(self, pckt):
        print( 'SensorNode',self.addr,'received a HELLO from', pckt[0])
        hellopckt = pckt[2]
        self.routeTable[hellopckt[1]] = dict(zip( ('ntHop', 'numOfHops', 'dstSeqNo', 'actvNgbrs', 'expirTime'),
(hellopckt[1], 1, hellopckt[2], [], 3 )))
        return True
    def Recv_RREQ_pckt(self, pckt)
        for t, entry in self.Received_Req_Q:
            if entry[1] == pckt[2][1] and entry[3] == pckt[2][3]:
                return False
        print ('SensorNode',self.addr,'received a RREQ from',pckt[0], 'which is', pckt[2][1], '->',pckt[2][4])
        if pckt[2][4] == self.addr:
            if self.routeTable.__contains__(pckt[2][1]):
                return True
            else:
                self.routeTable[pckt[2][1]] = dict(zip( ('ntHop', 'numOfHops', 'dstSeqNo', 'actvNgbrs', 'expirTime'),
(pckt[0], pckt[2][6], pckt[2][2], [], 3 )))
                self.pcktQ['snd'].append( [self.addr, pckt[0],['RREP',pckt[2][1],pckt[2][4],self.sequenceNo, 1,
'lifeTime']] )
            return True
        self.routeTable[pckt[2][1]] = dict(zip( ('ntHop', 'numOfHops', 'dstSeqNo', 'actvNgbrs', 'expirTime'),
                                    (pckt[0], pckt[2][6], pckt[2][2], [], 3 ))
        Try:
            if self.routeTable[pckt[2][4]]['dstSeqNo'] > pckt[2][5]:
                entry = self.routeTable[pckt[2][4]]
                self.pcktQ['snd'].append([self.addr, entry['ntHop'],['RREP',pckt[2][1],pckt[2][4], entry['dstSeqNo'],
entry['numOfHops'], 'lifeTime'] ] )
                return True
        except:
            pass
        tmppckt = pckt[2][:]
        tmppckt[6] += 1
        self.pcktQ['snd'].append( [self.addr, 'broadcast',tmppckt] )
        return True
    def Recv_RREP_pckt(self, pckt):
        print ('SensorNode',self.addr,'received a RREP from',pckt[0], 'which is', pckt[2][1], '->',pckt[2][2])
        if self.routeTable.__contains__(pckt[2][2]) == False:
            self.routeTable[pckt[2][2]] =  dict(zip( ('ntHop', 'numOfHops', 'dstSeqNo', 'actvNgbrs', 'expirTime'),
(pckt[0], pckt[2][4], pckt[2][3], [], 3 )))
                for index, entry in zip(range(sys.maxsize), self.Received_Req_Q):
            if pckt[2][1] == entry[1][1] and pckt[2][2] == entry[1][4]:
```

```
            print ('+--A successful Route Discorvery from', pckt[2][1], 'to', pckt[2][2])
            self.Received_Req_Q.pop(index)
        try:
            entry = self.routeTable[pckt[2][1]]
            tmppckt = pckt[2][:]
            tmppckt[4] += 1
            self.pcktQ['snd'].append( [self.addr, entry['ntHop'], tmppckt ] )
            print ('+--Found a route to',entry['ntHop'], 'for RREP from', pckt[2][1], 'to', pckt[2][2])
            return True
        except:
            print ('+--No route to', pckt[2][2], 'RREP from', pckt[2][1], 'to', pckt[2][2], 'dropped')
            return False
              def Recv_RERR_pckt(self, pckt):
        pass
    def BlockChain_Forma_Data(self, pckt):
        print (self.addr, 'DATA',pckt[2][1],'->', pckt[2][2], 'num:', pckt[2][3])
        strBlock=self.addr, 'DATA',pckt[2][1],'->', pckt[2][2], 'num:', pckt[2][3]
        np.save('ReceiveBlock.npy',strBlock)
        if pckt[2][2] == self.addr:
            return True
        else:
            if self.routeTable.__contains__(pckt[2][2]):
                pckt[1] = self.routeTable[pckt[2][2]]['ntHop']
                self.pcktQ['snd'].append(pckt)
            else:
                pass
        return True
    def BlockChainRecev(self):
        recivedpckt_dispatch=        {'HELLO':self.helloPktReceived,        'RREQ':self.Recv_RREQ_pckt,
'RREP':self.Recv_RREP_pckt,
                'RERR':self.Recv_RERR_pckt, 'DATA':self.BlockChain_Forma_Data }
            try:
            pckt  = self.pcktQ['rcv'].pop(0)
            if pckt[0] == self.addr:
                return
            if pckt[1] != self.addr and pckt[1] != 'broadcast':
                return
            if recivedpckt_dispatch[pckt[2][0]](pckt):
            self.stsQ['rcvd'][pckt[2][0]].append(pckt[2])
        except:
```

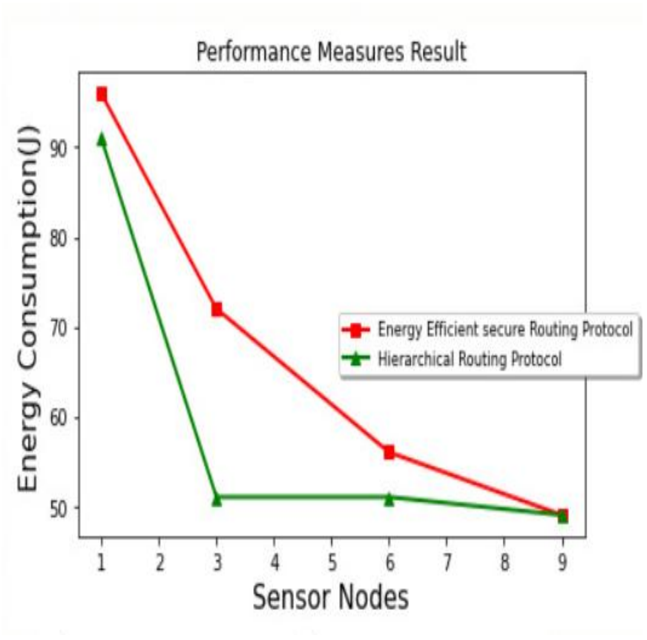
```
        return
class SensorNode:
    RoutingProtocol = False
    mMod = False
    def __init__( self,rp, addr, mm, currentnode_Pos, next_Pos, currentnode_Speed,  *arg):
        self.RoutingProtocol = rp(addr)
        self.mMod = mm(currentnode_Pos, next_Pos, currentnode_Speed, *arg
    def move(self):
        self.mMod.move()
    def Sender(self):
        return self.RoutingProtocol.Sender()
    def recive(self,pckt):
        self.RoutingProtocol.Receiver(pckt)
SensorNodes=  [SensorNode(HierClust,i,Assigning_random,  [random.randint(0,1000),random.randint(0,1000)]
[random.randint(0,1000),random.randint(0,1000)], 5, 1000,1000,5,5,20,1) for i in range(50)]
recivedpckt_dispatch = {'HELLO':'Hi', 'RREQ':'Send Request : 55', 'RREP':'Send Response message: 12',
                    'RERR':'Received Reply Message:68', 'DATA':[random.randint(100,1000) for i in range(0,5)]}
arrx = []
arry = []
for i in range(2000):
    for index, SensorNode in zip( range(sys.maxsize),SensorNodes):
        SensorNode.move()
        pos = SensorNode.mMod.currentnode_Pos
        arrx.append(pos[0])
        arry.append(pos[1])
PerfRes=np.load('existing.npy')
plt.plot(arrx[0:300], arry[0:300],'go')
plt.show()
figure, axes = plt.subplots()
plt.plot(arrx[0:100], arry[0:100],'go')
Drawing_colored_circle = plt.Circle(( 800,200), 200,fill = False)
 axes.add_artist( Drawing_colored_circle )
Drawing_colored_circle = plt.Circle(( 900,750), 200,fill = False)
axes.add_artist( Drawing_colored_circle )
Drawing_colored_circle = plt.Circle(( 550,500), 200,fill = False)
axes.add_artist( Drawing_colored_circle )
Drawing_colored_circle = plt.Circle(( 200,750), 200,fill = False)
axes.add_artist( Drawing_colored_circle )
Drawing_colored_circle = plt.Circle(( 200,275), 200,fill = False)
axes.add_artist( Drawing_colored_circle )
```

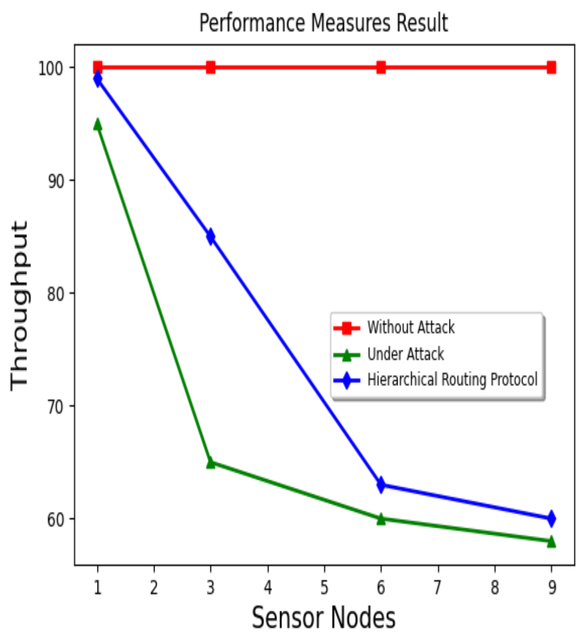
```
Drawing_colored_circle = plt.Circle(( 500,850), 200,fill = False)
axes.add_artist( Drawing_colored_circle )
plt.title( 'Colored Circle' )
plt.show()
SensorNodes[0].RoutingProtocol.pcktGeneration([0, 'broadcast',['HELLO',0,1 ]])
pkt = [0]
arrX=[]
arrY=[]
import numpy as np
for t in range(5):
    for index, SensorNode in zip( range(sys.maxsize),SensorNodes):
        pos = SensorNode.mMod.currentnode_Pos
        valx=arrx[index]
        arrX.append(np.array([valx,pos[0]]))
        valy=arry[index]
        arrY.append(np.array([valy,pos[0]]))

        SensorNode.RoutingProtocol.timer()
        SensorNode.move()    pkt.append(len(SensorNodes[31].RoutingProtocol.stsQ['rcvd']['DATA']))
    print ('Iteration:', t)
    for n in range(1):
        for netSensorNode in SensorNodes:
            i = netSensorNode.RoutingProtocol.addr
            pckt = netSensorNode.Sender()
            for SensorNode in SensorNodes:
                if i != SensorNode.RoutingProtocol.addr:
                    if distance(SensorNode.mMod.currentnode_Pos, netSensorNode.mMod.currentnode_Pos) <200:
                        SensorNode.recive(pckt)
                    else:
                        pass
AttackNodes=[1,3,6,9]
fig,ax=plt.subplots(1)
plt.plot(AttackNodes,PerfRes[0][0],'rs-',lw=2,label='Without Attack')
plt.plot(AttackNodes,PerfRes[0][2],'g^-',lw=2,label='Under Attack')
plt.plot(AttackNodes,PerfRes[0][1],'bd-',lw=2,label='Hierarchical Routing Protocol')
legend = ax.legend(loc='upper left', shadow=True, fontsize='small',bbox_to_anchor=(0.5,0.5))
plt.xlabel("Sensor Nodes", fontsize=15)
plt.ylabel('Throughput', fontsize=15)
plt.title('Performance Measures Result')
plt.show()
```

```
fig,ax=plt.subplots(1)
plt.plot(AttackNodes,PerfRes[1][0],'bd-',lw=2,label='Without Attack')
plt.plot(AttackNodes,PerfRes[1][2],'rs-',lw=2,label='Under Attack')
plt.plot(AttackNodes,PerfRes[1][1],'g^-',lw=2,label='Hierarchical Routing Protocol')
legend = ax.legend(loc='upper right', shadow=True, fontsize='small',bbox_to_anchor=(0.55,1))
plt.xlabel("Sensor Nodes", fontsize=15)
plt.ylabel('Average End-to-End Delay(ms)', fontsize=15)
plt.title('Performance Measures Result')
plt.show()
fig,ax=plt.subplots(1)
plt.plot(AttackNodes,PerfRes[2][1],'rs-',lw=2,label='Energy Efficient secure Routing Protocol')
plt.plot(AttackNodes,PerfRes[2][2],'g^-',lw=2,label='Hierarchical Routing Protocol')
legend = ax.legend(loc='upper left', shadow=True, fontsize='small',bbox_to_anchor=(0.5,0.5))
plt.xlabel("Sensor Nodes", fontsize=15)
plt.ylabel('Throughput', fontsize=15)
plt.title('Performance Measures Result')
plt.show()
fig,ax=plt.subplots(1)
plt.plot(AttackNodes,PerfRes[2][1]-np.array([random.randint(3,10)  for  x  in  range(0,len(PerfRes[2][1]))]),'rs-
',lw=2,label='Energy Efficient secure Routing Protocol')
plt.plot(AttackNodes,PerfRes[2][2]-np.array([random.randint(3,10)  for  x  in  range(0,len(PerfRes[2][1]))]),'g^-
',lw=2,label='Hierarchical Routing Protocol')
legend = ax.legend(loc='upper left', shadow=True, fontsize='small',bbox_to_anchor=(0.5,0.5))
plt.xlabel("Sensor Nodes", fontsize=15)
plt.ylabel('Energy Consumption(J)', fontsize=15)
plt.title('Performance Measures Result')
plt.show()
print('Received Packet')
print(recivedpckt_dispatch)
```

4. PERFORMANCE ANALYSIS





5. CONCLUSION

In this project, we present a secure authentication and routing mechanism for WSNs. The aim of our proposed mechanism is to carry out authentication of the sensor nodes and ensure the secure communication between the nodes and BS. The proposed routing protocol selects the nodes on the basis of shortest distance from the BS. Whereas, a secure authentication mechanism of nodes is performed using the Bidirectional-blockchain .Results show that our proposed model improves the packet delivery ratio and the network lifetime. In future work, the proposed idea will be tested on larger networks and a realistic routing environment.

6. REFERENCES

[1]. Y. Hu, M. Dong, K. Ota, et al. "Mobile Target Detection in Wireless Sensor Networks with Adjustable Sensing Frequency, "IEEE System Journal, Doi: 10.1109/JSYST.2014.2308391, 2014.

[2]. M. Dong, K. Ota, A. Liu, et al. "Joint Optimization of Lifetime and Transport Delay under Reliability Constraint Wireless Sensor Networks," IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 1, pp. 225-236, 2016.

[3]. S. He, J. Chen, F. Jiang, et l. "Energy provisioning in wireless rechargeable sensor networks," IEEE transactions on mobile computing vol. 12, no. 10, pp. 1931-1942, 2013.

[4]. X. Liu, M. Dong, K. Ota, P. Hung, A. Liu. "Service Pricing Decision in Cyber-Physical Systems: Insights from Game Theory," IEEE Transactions on Services Computing, vol. 9, no. 2, pp. 186-198, 2016

[5]. C. Zhu, H. Nicanfar, V. C. M. Leung, et al. "An Authenticated Trust and Reputation Calculation and Management System for Cloud and Sensor Networks Integration," IEEE Transactions on Information Forensics and Security, vol. 10, no. 1, pp. 118-131, 2015.

[6]. A.Liu, M.Dong, K.Ota, et al."PHACK：An Efficient Scheme for Selective Forwarding Attack Detecting in WSNs," Sensors, vol. 15, no. 12, pp. 30942-30963, 2015.

[7]. A. Liu, X. Jin, G.Cui, Z. Chen, "Deployment guidelines for achieving maximum lifetime and avoiding energy holes in sensor network,"Information Sciences,vol. 230, pp.197-226, 2013.

[8]. Z. Zheng, A. Liu, L. Cai, et al."Energy and Memory Efficient Clone Detection in Wireless Sensor Networks, "IEEE Transactions on Mobile Computing.vol. 15, no. 5, pp.1130-1143,2016.