A model for Underwater Security in Communication Using Secret Key Algorithm and Node Value

by

MD. Saidul Arifin Shuvo
17301122
Mustafa Tamim Firdaus
17301134

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
June 2021

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

MD. Saidul Arifin Shuvo
17301122

Mustafa Tamim Firdaus
17301134

Approval

The thesis titled "A model for Underwater Security in Communication Using Secret Key Algorithm and Node key" submitted by

1. MD. Saidul Arifin Shuvo (17301122)
2. Mustafa Tamim Firdaus (17301134)

of Spring, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on June 06, 2021.

Examining Committee:

Supervisor:
(Member)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

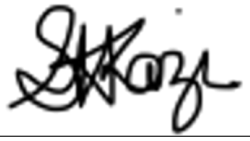
Co-Supervisor:
(Member)

Arif Shakil
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)



Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

We, MD. Saidul Arifin Shuvo, Mustafa Tamim Firdaus sincerely and consciously state that this paper "A model for Underwater Security in Communication Using Secret Key Algorithm and Node key" is our own contribution which has not been published anywhere. This thesis paper highlights the own work of thesis authors where we have been cited the contributions of other's work properly. Lastly, we want to publicize that, we give acknowledgement and credit to the persons from whom we took help completing our thesis work.

Abstract

Underwater wireless network has become one of the most significant research areas in the networking field nowadays as it has huge possibilities. Underwater communication can be used for collecting data of undersea ecology, secure naval area, pre alarm for natural disaster etc. Most of the surface in our world is surrounded by water; for this reason underwater communication can be hugely beneficial for us. However, secure the network and its database is a big challenge as this kinds of network is being setup remotely; far away from human touch. In this paper, we have introduced a new model of secure communication system which will secure the network and its database using AES secret key generation algorithm and node key.

Keywords: AES algorithm; Node key; AUV; Sensor; Mssql database; Security

Acknowledgement

At the very beginning, all praise to our creator the Great Almighty Allah for whom our thesis have been completed without any major interrupt.
Next, sincere gratitude to our supervisor Sadia Hamid Kazi ma'am and Arif Shakil sir for their full guidance and contribution for completing our thesis work. They always encouraged us to think innovatively and out of the box.
After that, we would like to direct our humble thanks to our supportive friends who have been there to give us mental support in these hard times.
At last, we want to take a moment to thank our parents as it may not be possible without their unconditional support and prayer.

Table of Contents

List of Figures

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

AES Advanced Encryption Standard

AUV Autonomous Underwater Vehicle

$CSMA$ Carrier Sense Multiple Access

DES Data Encryption Standard

$RDBF$ Relative Distance Based Forwarding

RSA Rivest Shamir Adleman

$SDRT$ Segment Data Reliable

$STUMP - WR$ Staggered TDMA Underwater MAC Protocol

$TDMA$ Time Division Multiple Access

$UW - ASN$ Underwater Acoustic Sensor Network

Chapter 1

Introduction

Underwater wireless communication system [1] is basically a communication platform that put into consideration below surface along with acoustic and optical wireless communications. This communication system is one of the most focused points of the networking area though it has some drawbacks because of the unique characteristics of the underwater environment. The unique characteristics of this environment has some protocols that has been proposed like CSMA (Carrier-sense multiple access), CDMA (Code division multiple access), TDMA (Time division multiple access), two spread-spectrum physical layer techniques for MAC protocol, Localization scheme for routing protocols, SDRT (Segment Data Reliable) using Torando codes for transport layer protocol, cross layer protocol etc. Again, STUMP-WR protocol was also introduced to lenient the underwater communication system. After that, an energy efficient MAC protocol was also introduced specially for underwater behaviors. Another thing is that securing the underwater communication was also a big concern here as the data collected from the medium which is very much sensitive and important in various aspects. However, most of the protocols and ideas introduced in research papers are mainly built on the idea of grounded communication where in reality mainly they are not actually perfect or workable on such a unique medium like underwater. As a result, all those proposed ideas are not actually helping underwater communication. For example, in a localization scheme we have to depend on GPS to correctly identify the node's location, but radio frequencies do not give accurate results underwater. In addition, because of identical behavior of underwater, those protocols do not give any accurate result.
Moreover, we want to propose such a secure model for underwater wireless communication where every single data will be preserved and transmitted in secure way. We have been come up with a more secure and hard to break secure key than the traditional AES secret key. The major contributions and topics of this paper are stated as below:

- **Novelty:** We have introduced a new secret key which is a combination form of AES algorithm's secret key and node data key. Again, we have added some customized header file for our work.
- **Performance:** We have achieved a better secure and length secret key compared to existing secret key algorithm model.

1.1 Research Problem

Managing the Underwater Network is more difficult than Surface Network. There are lots of characteristics of water to tackle and build a long lasting network. Nowadays security has become a major issue in the network area. We have to keep an underwater network safe along with a surface area network. Some people have worked on this matter but there is always an occurring problem to develop the idea. Limited bandwidth, variable propagation delay, limited signaling property, power consumption, real time data sampling and high memory space for caching data etc. all of these problems have occurred in different research papers.
For an underwater environment the MAC protocol is not suitable because of unique characteristics of underwater like finite bandwidth, very high propagation delays as well as variable, and towering error rates of high bit, disconnectedness behavior, channel asymmetry and heavy multipath-fading phenomena. All the introduced protocols of authors have some issues with underwater environments like creating a path to modified and updating network topology in proactive protocol, loading upper secrecy in creating paths in reactive and for geographical protocol obstacles in connecting with GPS in underwater. Again, authors describe that. In underwater the TCP implementation does not work properly because of propagation delay on the end to end path and for that round trip time (RTT) is hard to set.[2]
Secondly, authors describe that while applying or implementing MAC protocol in their proposed application some collisions occur like transmit-receive collisions and receive - receive collisions for unique characteristics of underwater.
In [3] the authors have described the security element for underwater water environment UWCN. Underwater communication is very much sensitive because of differences with ground based sensors like radio waves, large propagation delay, and low bandwidth. For this reason there's a huge possibility of being affected by malicious attacks. The authors describe the security system underwater by sensors and autonomous underwater vehicles or AUV is very much challenging due to activity of those AUV's and movement of sensors with water currents in underwater. The authors also said that, though geographical routing is committing for underwater networks due to scalability limited signaling properties, we cannot rely on it fully as radio waves do not work on underwater properly. After that, the authors describe some of the attacks on UWCN like jamming, wormhole attack, sinkhole attack, hello flood attack, acknowledgement spoofing, Sybil attack etc.
In [4] authors Almir Davis and Hwa Chang describe most of the applications they proposed have to go through some challenges like power consumption or power harvesting. All the sensors or AUV's which are used to make surveillance undersea need to be recharged after a time period but in such an environment like underwater it's quite tough to maintain. Another important challenge that authors emphasize on is storage shortage. Additionally, sensors those have used underwater needed for a high memory space for caching the data which is also a big challenge. After that, because of the slow intermittent medium of underwater it's tough to maintain the real time data sampling and transmission.
Authors describe in their paper [5] deploying wireless networks in the underwater environment creates significant challenges as radio waves generate poor propagation assets under the surface means water but optical waves are extremely impacted by dispersion and need a high rate of accuracy in pointing the laser beams.

1.2 Research Objectives

The goal of this research is mainly to develop a secure network for underwater which has some unique characteristics. Underwater environment has got some unique characteristics and for that we cannot apply all the methods or protocols that we use in surface models. That's why underwater networks get affected by outsiders so often. To reduce this attack our objectives through this research is to develop a secure model where all devices will connect to the network after verifying the secret key of each device which will be generated by the network.

1. To deeply understand the underwater environment, and its unique characteristics.
2. To deeply understand the wireless network underwater and how it works.
3. To deeply understand the underwater wireless network's attacks by outsiders.
4. To develop a suitable secure model for an underwater network.
5. To appraise the model.
6. To deliver directions and suggestions on enhancing the model.

Chapter 2

Literature Review and Related Work

2.1 Underwater Network's Protocol and Application

2.1.1 UW-ASN mac Protocol

A related study of underwater wireless networks the authors [2] Dario Pompili and Ian F. Akyildiz describes the overview of recent medium access control, routing, transport, and cross layer networking protocol. Considering unique characteristics of underwater MAC solutions for underwater is mainly focused on carrier-sense multiple access (CSMA) and code division multiple access (CDMA). CSMA or carrier-sense multiple access is a MAC protocol where data transmitted after checking if the shared medium is free or not in a connecting place. On the other hand, in CDMA data transmitted in a shared line simultaneously where but all the sender or stations have different code to generate data and receive data. After that, the authors also proposed UW-MAC, a distributed single carrier CDMA solution which is made for achieving mainly three goals in deep water communication- high network throughput, low channel access delay and low energy consumption. In UW-MAC protocol, available connected nodes do not need to use the same transmission power synchronized. Again, the time frame divides in two subparts, where one part is used to send data using time division multiple access (TDMA) and another part is used to accommodate traffic related variable conditions for unscheduled access. Secondly, the authors highlighted the currently built routing protocols by dividing them into three sections - proactive, reactive and geographical protocols. To avoid the obstacles for underwater characteristics, authors proposed a localization scheme which introduced a non-degenerative projection technique of 2D problem which reduces the 3D localization problem. In this proposed model, there are d+1 anchor nodes needed in d dimension to correctly localize a network area. Based on this projection in a 2D plane containing nodes to be localized, nodes can be localized successfully. In routing protocol authors also cited from other research that, if the number of hop can be increased then it will improve the attainable information rate and credibility. Thirdly, the authors describe the obstacles and challenges of transport layer protocol in current existing protocol. As in underwater TCP implementation does not work properly challenges the author describes how a transport

layer protocol for underwater should be organized for example shadow zones can be handled by pre determining losses of connectivity, the energy consumption can be minimized through SACK or selective acknowledgement, relying on rate-based transmission of data, properly handling out of sequences packet forwarding, finding information from lower layers to predetermine etc. Considering all those criteria a transport layer protocol was proposed Segment Data Reliable Transport (SDRT) where error packets will be recovered by using Tornado Codes. In last, considering all the problems, obstacles and challenges only in the underwater environment the authors suggest an effective solution called Cross Layer Protocols which will enable the feasible use of resources to network performance and avoid duplication of functions.

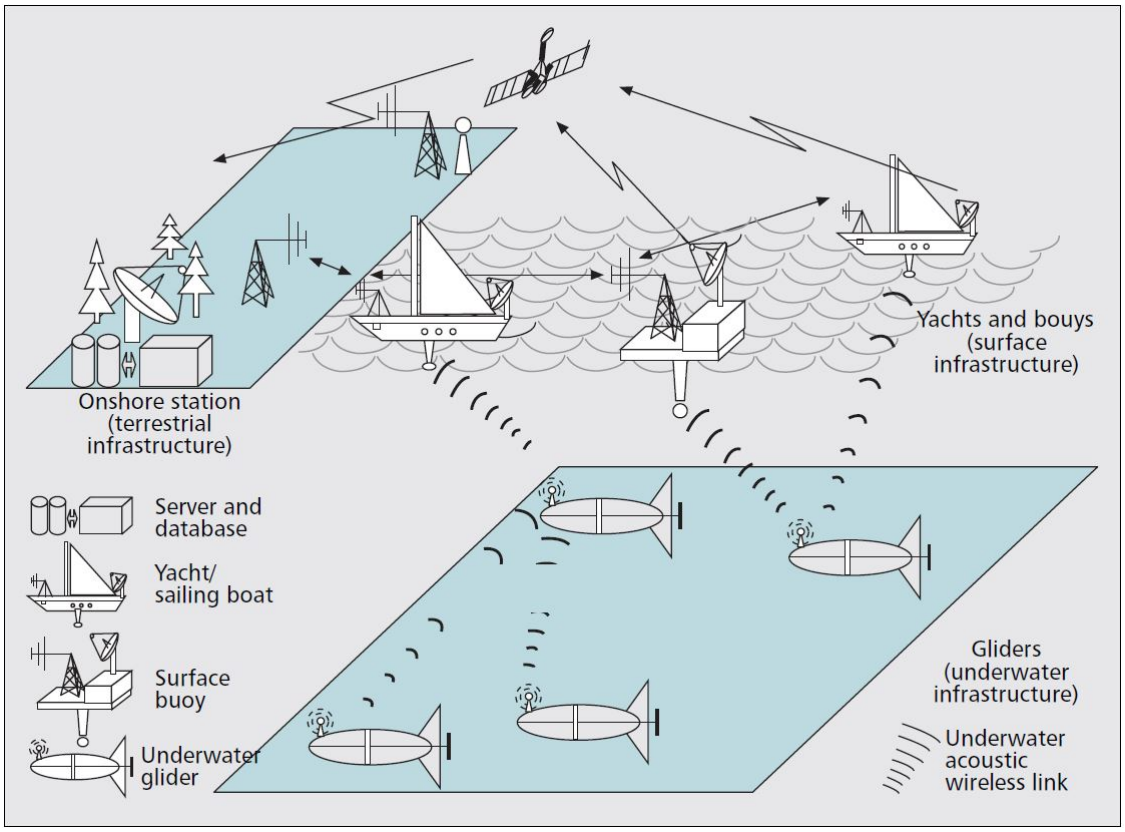


Figure 2.1: UW-ASN composed of underwater and surface vehicles [2]

2.1.2 STUMP-WR routing Protocol

Authors Kurtis Kredo and OrasantMohapara in their paper [3] come up with a new solution named STUMP-WR, a distributed routing protocol and channel scheduling protocol which reduce some unique problem in under water like long propagation delays and achieving higher throughput and lower energy consumption per bit delivered. The STUMP-WR protocol improves under water network performance by gaining the knowledge of delays which is the reason for overlap between multiple transmissions. In this protocol, a distributed algorithm is used to select and schedule links using local state knowledge. For selecting the link, node selected the link

with minimum hop count and maximum remaining energy. In this way, total energy consumption got reduced as well as the network lifetime got extended. After that for scheduling the link, nodes can schedule links with the knowledge of Long propagation delay to minimize the idle time as well as reduce overhead. Next, in a network model for underwater the authors describe possible link conflicts like TX-TX, TX-RX, RX-RX, TX-RX-TX among nodes depending on individual scenarios. For decreasing these conflicts, the authors describe the idea of DSSS CDMA radio where node transmission will execute by code sequences assigned to nodes and multiplexing the facilities various signals to occupy a single transmission channel. After that, ensuring the conflict free scheduling the author presented the idea of finding the first transmit slot s_i and p_i for finding propagation delay. By finding these parameters, collisions got reduced by determining the required duration, i for interfacing the link. For finding the valid schedule, scheduler constraints will be placed at the beginning slot of each link. For example the authors showed a conflict type TX-RX constraint between link i and j where destination of j is the source of i. The constraints for deciding the conflicts are:

$$s_i \geq s_j + \Delta i + \frac{p_j}{T} \tag{2.1}$$

$$s_i \geq \frac{p_j}{T} + m \geq s_i \, \Delta i \tag{2.2}$$

where p is the propagation delay

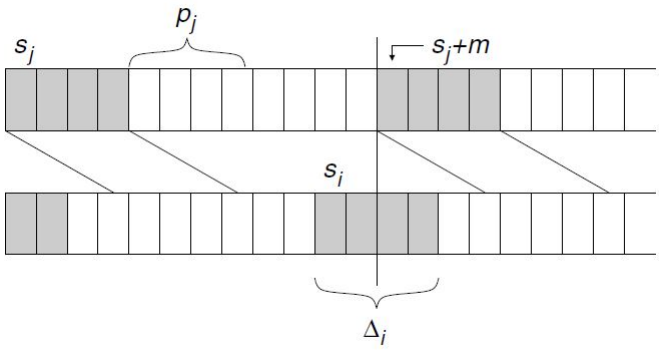


Figure 2.2: Schedule Constraint for a TX-RX conflict[3]

Constraint equation for each conflicts,

$$\Delta i - pjT + m \leq sj - si - moij \leq -\Delta j - pjT, \; (for \, TX - RX) \tag{2.3}$$

$$\Delta i + pi - pjT - m \leq sj - si - moij \leq -\Delta i + pi - pjT, \; (for \, RX - RX) \tag{2.4}$$

$$\begin{aligned} \Delta i + pi - p(\text{source j, destination i})\, T - m \leq sj - si - moij \\ \leq -j + pi - p(\text{source j, destination i})T, \; for \, (TX - RX - TX) \end{aligned} \tag{2.5}$$

$$\Delta i - m \leq sj - si - moij \leq -\Delta j, \; (for \, TX - TX) \tag{2.6}$$

In the article author's added scheduling algorithm so that nodes can attained a valid scheduler through ordering variables oij

1: $m_{min} \leftarrow 0$

2: **for all** $i : src_i = n$ **do**

3: $LB \leftarrow 0$

4: $UB \leftarrow m - 1$

5: **for all** $j \in L_i : i \neq j$ **do**

6: **if** $o_{ij} = 0$ *then*

7: $LB \leftarrow max\{ LB, s_j - \overline{B_{ij}} \}$

8: $UB \leftarrow min\{ UB, s_j - \underline{B}_{ij} \}$

9: **end if**

10: **end for**

11: $m_{min} \leftarrow max\{ m_{min}, m - UB + LB \}$

12: **end for**

13: **return** m_{min}

Figure 2.3: Schedule Algorithm[3]

In last, authors have shown their results of using STUMP-WR protocol by achieving the efficiency, avoiding energy losses of packet collisions in a network.

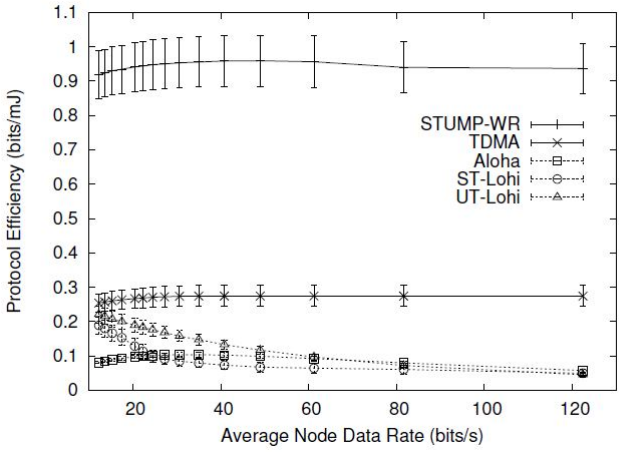


Figure 2.4: Protocol Efficiency[3]

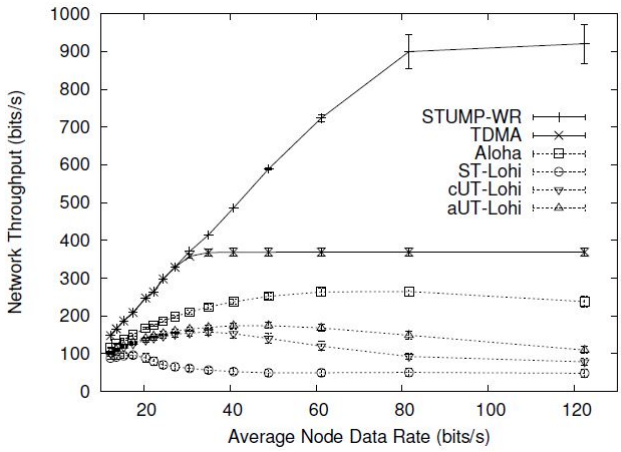


Figure 2.5: Protocol Throughput[3]

2.1.3 UWAN delay tolerate MAC protocol

In another article[6] VolkanRodoplu and Min Kyoung Park introduced another distributed, scalable and energy efficient MAC protocol named UWAN MAC protocol for delay-tolerate application like underwater ecological sensor network where energy is the main performance concern instead of bandwidth utilization like ALOHA, MACA, MACAW protocols. Authors have concentrated on such a protocol which will work under large propagation delays in a deep network of hundreds sensors especially for ecological monitoring application. For such a dense network battery life is very important that's why the authors proposed the idea of sleep mode so that energy can be saved with a low duty cycle. In this proposed MAC protocol locally synchronizes the schedule even in long propagation delay through determining the listen cycle where a node's transmission cycle T is declared by preamble beacon signal. Then, at the time of joining a new node to the network for frame synchronization it checks preamble sequence and find the length of transmission cycle period.

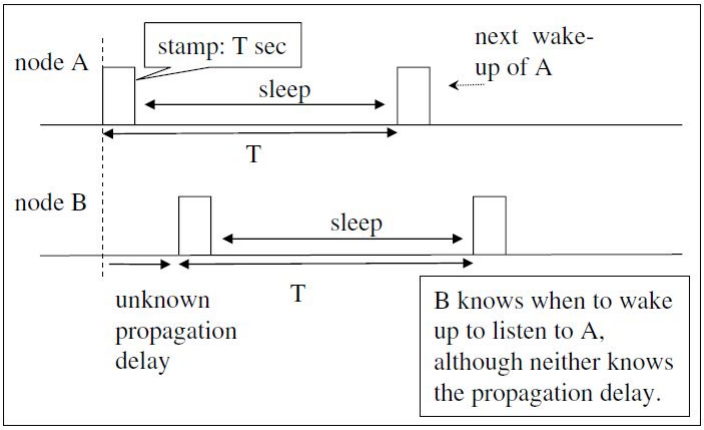


Figure 2.6: Basic idea of UWAN MAC Protocol[6]

In this way the new node knows when to listen to the previous node. However, authors describe that in this application some collisions may occur like transmit-receive collisions and receive - receive collisions. For initializing MAC protocol, every node broadcasts its SYNC packet or beacon signal by waking till the first cycle. In addition, through this signal neighbor nodes can find out the transmission cycle and after what times it will transmit data. In this way through decoding the SYNC packets each node can determine its sleep schedule time.

2.1.4 RDBF routing protocol by calculating fitness factor

In the next paper[7] we have found that they are using relative distance-based forwarding (RDBF) routing protocol which gives communication systematic, power reduction and minimum detain routing. According to the author, they're using a fitness factor to calculate and evaluate the degree of suitability for the packets to be forwarded by a node. Author emphasized that under the limits of the RDBF will restrict the reach of the nominee forwarders and discover the beneficial relays to forward packets in the fitness factor. In addition, Relative distance-based forwarding (RBDF) is always trying to search an ideal routing path to the sink node from the source node. To calculate the efficiency of a node to be the rely node and limit the number of nodes participating in the forwarding process, they employed a fitness factor. According to the author's result, RDBF only requires nodes that have fitness factors that are under a threshold to involve in packet routing. In addition to balance energy consumption they take the retained data from the sensor nodes into consideration. Relative distance-based forwarding (RDBF) also commands the communication period of the time of different multi forwarders to lessen the corresponding identical of the package of data. The simulation results author works show that the efficiency of the RBDF is superior to the well-known node density and mobile scenarios. Lastly Author added that VBF is location-based routing protocol based on the average end-to-end latency protocol and aspects of packet transmission ratio. Again, power performance of the RDBF is approximate to the VBF.

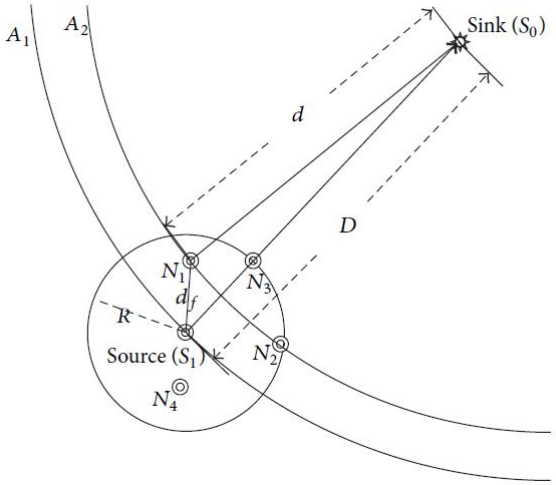


Figure 2.7: Fitness factor forward[7]

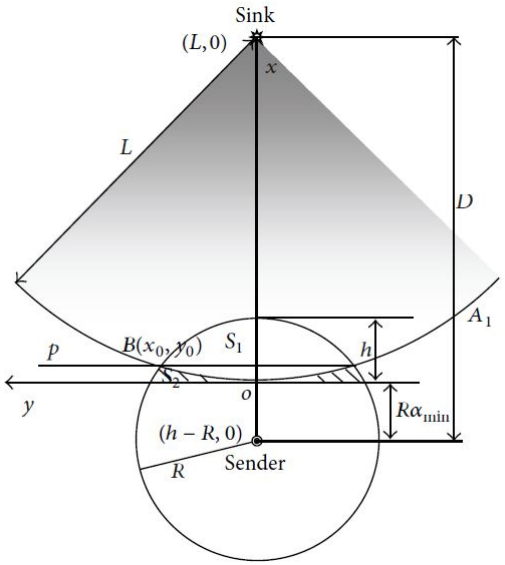


Figure 2.8: The region of the candidate[7]

2.1.5 Hybrid approach of UWSN application

In [4] authors Almir Davis and Hwa Chang describe the hybrid approach underwater wireless sensor networks (UWSN) where they mainly discuss UWSN's applications and their challenges. There are several applications of UWSN like monitoring applications, seismic application, military and homeland security application, disaster prevention disaster recovery application.

2.1.6 System based data circulation in UWAN

Next, authors [5] proposed a system based on spreading or circulation of data in underwater acoustic wireless networks. According to Author, their suggested frame-

work efficiently fights against the problem of high latency of the underwater acoustic wireless environment. From their final simulations result, they stated that Simulation findings are superior efficiency of the proposed system in the aquatic nature relative to modifications to current terrestrial pressure systems. Furthermore, the production of the suggested mechanism is not impacted by growing the broadcast server's communication range. Author solved their problem of deploying wireless networks in underwater implementing variants of acoustic protocol such as Aloha, medium access with collision avoidance (MACA), and floor acquisition multiple access (FAMA) medium access control (MAC).

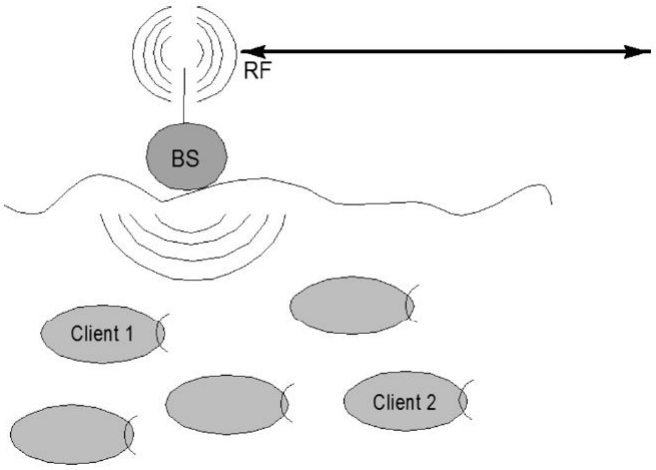


Figure 2.9: Underwater push system comprising a base station with acoustic transmission capabilities and a number of clients[5]

Then, Author claimed that [8] because of its huge influence on the production of the network, considerable attention has been drawn to Medium Access Control (MAC). In comparison to the earth networks, which necessarily depend on radio waves for communication, UWSNs use Acoustic waves, which present a new obstacle to research in architecture of the protocols of the MAC. In the initial stages, the efficiency of the UWSNs in terms of latency and throughput was the big priority of the MAC layer protocol architecture. Author mentioned some of the applications such as Ocean sampling networks, Environmental monitoring, Disaster prevention, assisted navigation, Distributed tactical surveillance, Mine reconnaissance etc. can be developed by their proposed system. In addition to that point, in order to deliver maximum performance in an energy-efficient manner, the implementation of an efficient Medium Access Control (MAC) protocol for UWSNs is of vital importance since the MAC layer protocol manages the connection of nodes to the mutual wireless channel. The MAC layer protocol in UWSNs seems to be of high significance to the network use in the existence of a severe acoustic underwater channel. Finally, the author concluded by saying that, Future study is likely to examine the production of various MAC protocols and localization algorithms. In addition, the cross-layer architecture of MAC protocols and localization problems should be examined.

2.2 Secret Key Algorithms

2.2.1 RSA

RSA or Rivest Shamir Adleman algorithom was [9] first introduced by two cryptographer Ron Rivest- Adi Shamir and one computer scientist Leonard Adleman. In 1978, these 3 persons invented an asymmetric cryptosystem or public key cryptosystems which is a block cipher system based on prime number theory. As it is an asymmetric secret key; that's why it use to two different key for encryption and decryption. Additionally, RSA algorithm use 2 prime numbers for generating the public - private key for encrypt the data and decrypt the data. However, there are mainly 3 steps [10] in RSA algorithm process and they are key generation, encryption and decryption. Above all, RSA algorithm has some limitations like if a small value of two prime numbers are chosen for generating the keys then it becomes too weak and can be decrypted easily. Similarly, if a bigger values are chosen then the processes take more time to execute. Again, this algorithm needs to set a same lengths for two prime which is also difficult to maintain. For these drawbacks, RSA is not actually ready for mercantile uses.

Public - Private Key Generation:

- 2 random prime number p and q are choose which are comparatively large where p is not equal to q.
- Calculate $n = pq$
- Enumerate: phi (n) = (p -1) (q -1)
- Choose a integer value which is $1 < e < phi(n)$
- Calculate d where d is kept as private key exponent and d $\times$ e = 1 mod phi (n);
- Public key will be (n , e) and the private key will be (n , d) where all three value kept d, p, q and phi secret.

Encryption Process:

- $Plaintext = P < n.$
- $C = P^e mod n$, in here C indicates the cipher text

Decryption Process:

- Cipher text = C
- $P = C^d mod n$, in here P indicates the plain text

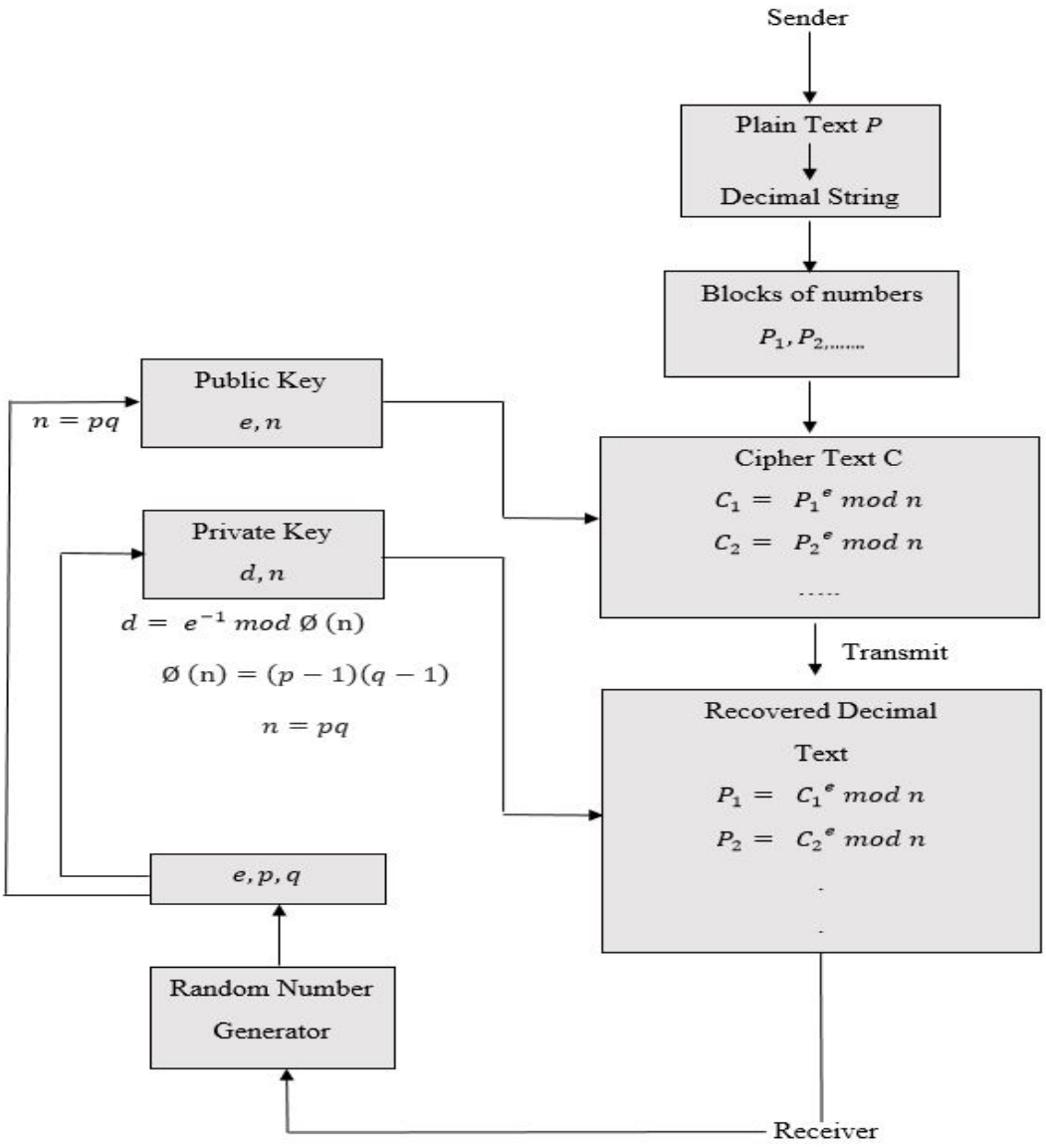


Figure 2.10: RSA processing of Multiple Blocks [10]

2.2.2 DES

DES or data encryption standard was proposed in 1970s first by IBM; but later on it was developed by NIST (National Institute of Standard and Technology). DES secret key algorithm is one of the famous cryptographic system which is a block cipher meant to decrypt data blocks of 64 bits. Additionally, DES use 64 bits of key to decrypt 64 bits of data blocks every time. [10] However, DES is actually key is 56 bits in length and the right most bits are use as parity bit to fit odd number of 1s. In actual process those parity bits are ignored and 7 bits are actually used in every byte. After that, DES use total sixteen iterations to mixing the plain text with key. As DES is a symmetric type of secret key, it used one same key for encryption decryption the data blocks. Though DES algorithm is used in many commercial use but it has affected by many vulnerable attacks from the beginning which indicates its weaknesses.

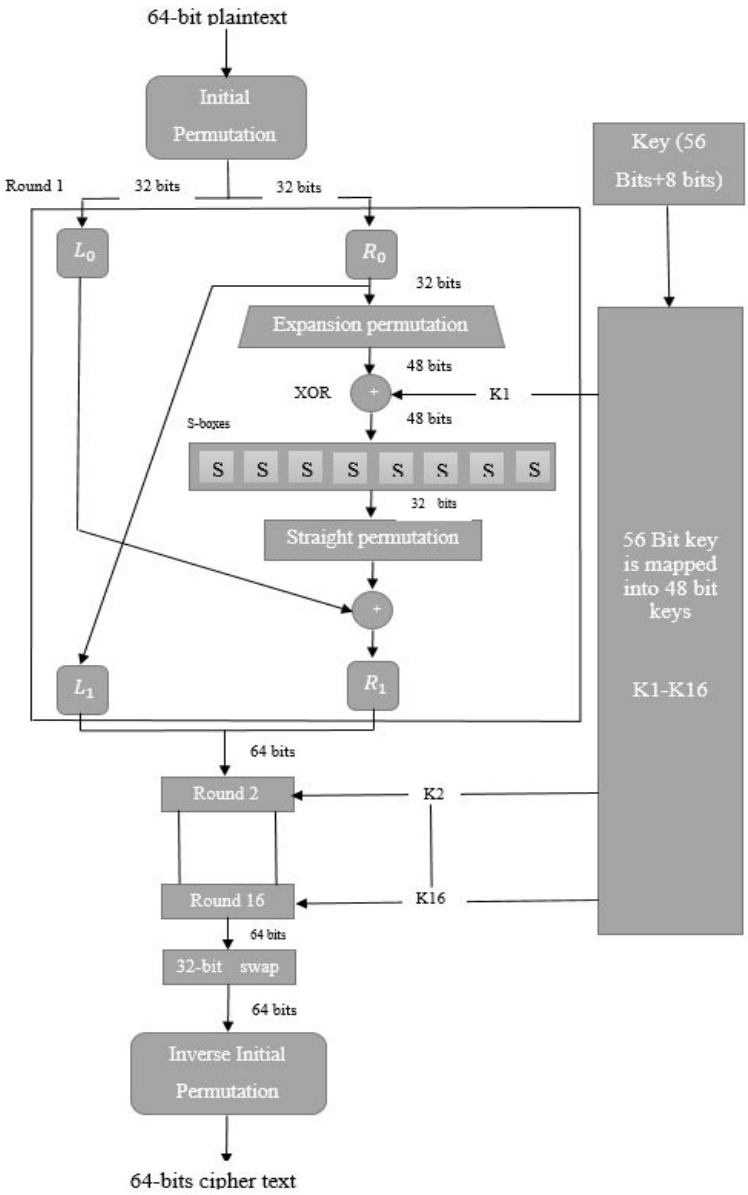


Figure 2.11: General Decryption of DES [10]

2.2.3 AES

Advanced Encryption Standard or AES is one of the most secure secret key algorithm which is replaced by DES algorithm in 2001. Recommended by National Institute of Standards and Technology this algorithm can clinch up to 128 bits of data with any combination under the limit. Additionally, there are three length of key are available in AES algorithm (AES-128, AES-192, and AES-256). Moreover, this key allocations are given with the help of several rounds (10 rounds for 128 key length, 12 rounds for 192 key length,and 14 rounds for 256 key length) for generating cipher text and retrieve the original plain text. In AES algorithm, it basically divide the whole 128 bits of data into 4 blocks which are basically act as array of 8 bits where it arranged as 4×4 matrix known as state. The process of AES algorithm starts with[10] AddRoundKey stage for both encryption and decryption. Next, the resultant text goes by nine different round before it reach the final stage and

each round accomplished 4 conversion (Sub-bytes, Shift-rows, Mix-columns and Add round Key). Besides, in final round the 3rd operation do not take place. On the other hand, decryption process used the reverse operation and inverse function of encryption process to getting the plain text.

Conversion process of 'substitute-byte'

In AES algorithm, each data blocks has 16 bytes of data from 128 bit data block where each block of data byte (1 byte = 8 bit) is converted to other data block. In this stage, Rijndael 8 bit of substitution box is being used

Conversion of 'shifting rows'

In shifting rows conversion, bytes are basically get transposal where last three rows from the state get transferred cyclically based on the location of rows. For example, 1 byte will shift circularly for 2nd row and for 3rd row 2 bytes will get shifted.

Conversion of 'mixcolumns'

In this stage the bytes are expressed as polynomial forms after multiplying a fix matrix to every column vector. Additionally, this stage is coequal to matrix multiplication of every column that stays in the state.

Conversion of 'AddRoundKey'

This conversion is inverse operation of its own where present state's 128 bits round key's 128 bit get through XOR operation bitwise.

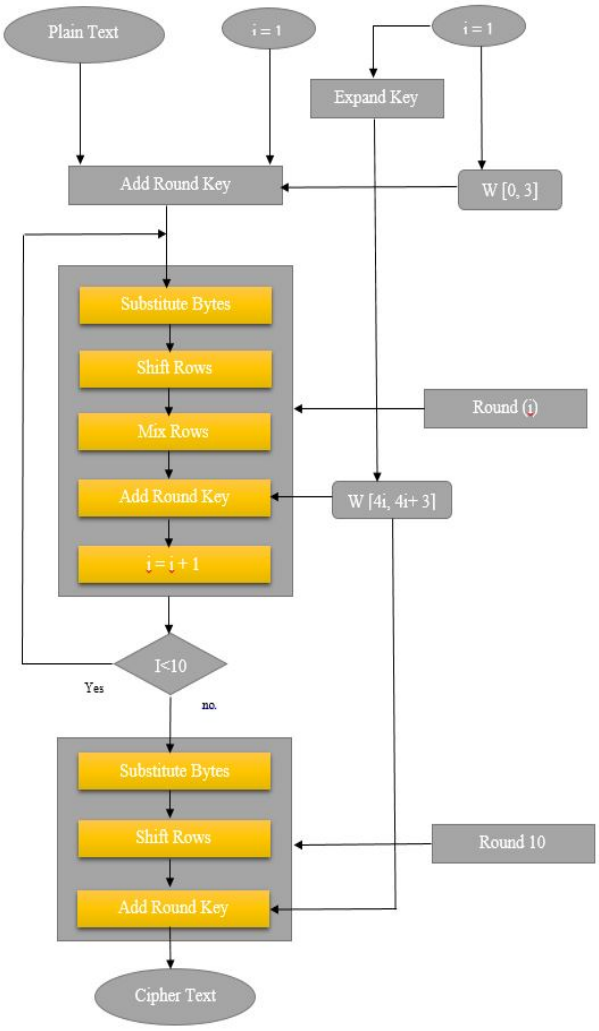


Figure 2.12: AES process [10]

Comparison
Key Length:

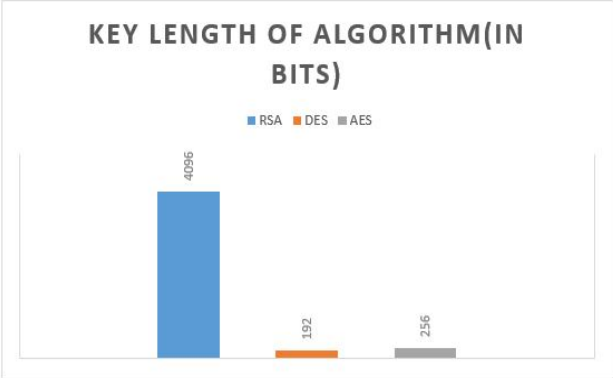


Figure 2.13: Column chart of key length of algorithms

Rounds:

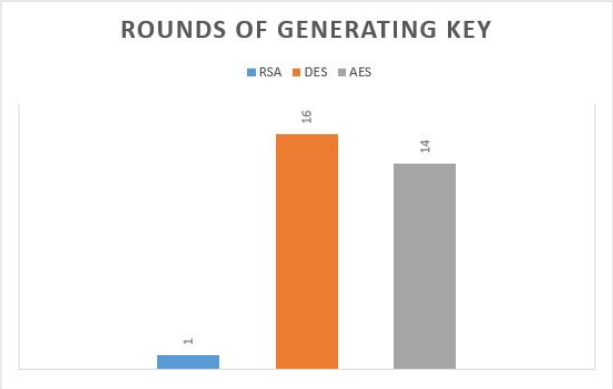


Figure 2.14: Column chart of rounds of algorithms

Plain Text:

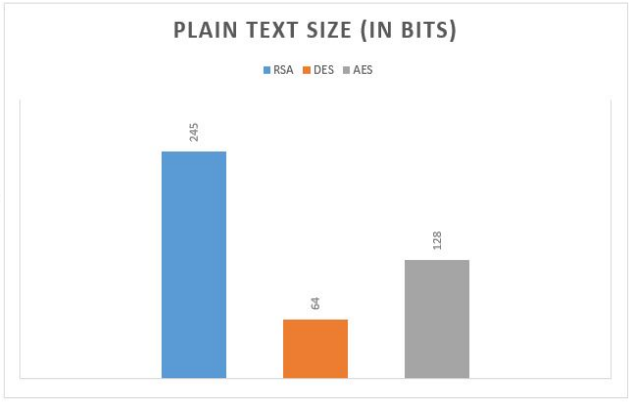


Figure 2.15: Column chart of plain text size of algorithms

2.3 SOLID Mechanism

SOLID is a design principle for object oriented programming where there are some set of standards for making class structure. SOLID principle is an essential at the time of designing phase of a project or software for calculating the feasibilities of the project. This principle help designer to avoid some factors like inflexibility, frailty, constancy and density for turning aside the redesigning. For managing these factors Robert C. Martin who is an American software engineer and instructor flourished the idea of SOLID in his paper called "Design Principles and Design Patterns". There are 5 basic principle in SOLID mechanism:

1. Single Responsibility Principle (SRP)
2. Open Closed Principle (OCP)
3. Liskov Substitution Principle (LSP)
4. Interface Segregation Principle (ISP)
5. Dependency Inversion Principle (DIP)

2.3.1 Single Responsibility Principle

This principle states that every class in the project should be changed for one reason. That means, every class of the project will be perform one task and in the same way methods used in the class should be limited for avoiding BLOB (Binary Large Object) class or multi responsibility class. When public methods are inexistent, then the class cannot performed any activity. Likewise, condition $c \epsilon C$ that conciliated the first principle of SOLID mechanism:

$$c \, \epsilon \, C, \; n(B_c^{public}) \, \epsilon 1, 2, ..., 9 \tag{2.7}$$

$$P\Big(c, n(B_c^{public})\Big) = (c \, has \, n(B_c^{public}) \, public \, methods) \tag{2.8}$$

For Example:

```
#include <iostream>
#include ''Node.h''
Using namespce std;
classChecking_Neighbour{
public:
intSecret_Key;
intNode_ID;
stringNode_Name;
voidvalidate_Neighbour(Node* temp){
if (temp!= NULL){
cout<< ''Node exists with key value in the Neighbour
list '' <<endl;
cout<< '''' <<endl;
else {
cout<< ''Node does not exists in the Neighbour list''
<<endl
cout<< '''' <<endl;
          }
   }
};
```

This class here indates the sungle tasking class where the "*Checking_Neighbour*" is only taking care of checking the neighbor list of newly coming nodes into the system through *validate_Neighbour* method. This class is not responsible for doing any other job of the project.

2.3.2 Open Closed Principle

The second principle of SOLID basis is entities of software (classes, modules, functions) will free for development or extension, but they will shut for editing. If one change in the project make occur cascade changing in the project, then the project is in bad shape. For that reasons, we should never change the entities; rather than we should extend the characteristics and behavior of the entities.

$\exists c \, \epsilon \, C$ Where there is one reference to the interface and denoting such interfaces like $(c) \, \epsilon \, I$:

$$\exists c \, \epsilon \, C, \; \exists i \, \epsilon \, I, \tag{2.9}$$

$$F(c)^{aggr} = F(c) \cup i \tag{2.10}$$

$$P(F(c)^{aggr}) = (F(c)^{aggr} exists) \tag{2.11}$$

$$I(c) = \{i \mid P(F(c)^{aggr}) = true;\ c \epsilon C,\ i \epsilon I\} \mid I(c) \mid\ > 1 \tag{2.12}$$

To illustrate:

```
Public class Circle : Shape
{
      Public double Radius {get; set;}
      Public override double Area()
      {
               Return Radius*Radius*Math.PI;
      }
}
Public double Area(Shape[]shapes)
{
        Double area = 0;
foreach (var shape in shapes)
        {
        Area += shape.Area();
         }
         Return area;
}
```

In here the area methods is only logic to looping the shape, calling the area method in this specific shape and return area. Moreover, if we want to add more new shapes then we do not have to change or edit the Area method. So that, the area method is close for modification but at the same time shape is open for extension as we will add new shape by defining the area method inside it.

2.3.3 Liskov Substitution Principle

In 3rd principle of Robert's development model is the liskov substitute principle where it shows that unnecessary use of inheritance in object oriented programming for getting escape from writing the code. This process is not appropriate in all situation as it may pass wrong value to the inherited class. For example:

```
Class Rectangle{
voidsetWidth(double w)
voidsetHeight(double h)
doublegetHeight()
doublegetWidth()
}
Class Square{
voidsetWidth(double w)
voidsetHeight(double h)
doublegetHeight()
doublegetWidth()
}
```

If square class extends rectangle class then both height width will set to w for setWidth and both height width value will set to h for setHeight. Then, if someone wants to test a rectangle

```
void test(Rectangle r){
r.setWidth(5);
r.setHeight(4);
assertEquals(5*4, r.getWidth() * r.getHeight())
}
```

In this case Square class will not pass the correct value and both setHeight and setWidth's value will set as 4. As a result it will give a wrong answer to assertEquals. So we should always inheritance only when we can replace the superclass by its sub class.

2.3.4 Interface Segregation Principle

In this principle the author states that, whenever two classes are communicating with each other; then their interfaces should be minimum. That means, there should not any unnecessary interfaces among them as they may cause a grubby circumstances. Again, ISP or interface segregation principle suggest that, several groups of methods can be made from the interfaces of the class and these groups will be used by different clients. So that, we can say in this principle every class that inherits interfaces should not have any bare methods.

$$\exists c \, \epsilon \, C, \, \exists i \, \epsilon \, I, \tag{2.13}$$

$$F(c)^{inh} = F(c) \cup i \tag{2.14}$$

$$B_c^{public} = \{\beta_c^{public} \mid \beta_c^{public} \neq \varnothing\} \tag{2.15}$$

```
Animal
void feed(); //abstract
void groom(); //abstract
Dog extends Animal
void feed(); //implementation
void groom(); //implementation
Tiger extends Animal
void feed(); //implementation
void groom(); //Dummy implementation
```

If I want to add groom method in the animal interface, then the Dog Tiger class also need to implement the groom method. The groom method should be applicable on domestic Animal and Dog but not Tiger class. So that, for keeping the compiler stable we have to run dummy implementation on Tiger class which is not good implementation. In this case, a standard process can be:

```
Animal
void feed(); //abstract
Pet extends Animal
void groom(); //abstract
Dog extends Pet
void feed(); //implementation
void groom(); //implementation
Tiger extends Animal
void feed(); //implementation
```

For avoiding this situation, we can create another interface Pet where we can implement the groom method after extending the Animal. Again, Dog can implement groom method as it is applicable for him by extending the Pet. On the other hand, Tiger will extends Animal, not Pet as it does not need to implement groom method.

2.3.5 Dependency Inversion Principle

Dependency Inversion Principle or DIP states that, modules of high level should not depend on modules of low level; rather then they should depend on abstractions among them. In addition, abstractions should be depend on details of the problem, but details should not depend on abstractions. For maintaining these scenario, we can established a abstractions layer between the high level and low level classes by isolating both the modules as well as isolating the abstractions from the detail. DIP design standard look at one structural feature and that is three alternative of operable of c belongs to C should be extended by any one of them.

$$(c^a \, \epsilon \, C, \; c^| \, \epsilon \, C, \; i \, \epsilon \, I) \tag{2.16}$$

$$F(c)^{acc} = \begin{cases} F(c) \cup F(c^a) \\ F(c) \cup i \\ F(c) \cup F(c^|) \end{cases} \tag{2.17}$$

2.4 Programming Language

2.4.1 C

Pros:
• Great for reverse-engineering and finding vulnerabilities.
• For tasks like networking, image processing or crypto program this language is lean, flexible, and efficient.
• It produces a huge energy to the computer's functions as it is a low-level language.
• Hackers cannot easily exploit this language if it is set with protection.
• Doesn't support classes and objects.

Cons:
• If not well protected, hackers can easily exploit.
• Doesn't support classes and objects.

2.4.2 C++

Pros:
• It is the updated version of C programming language and it's basically allows the unsupported classes objects of C++.
• Its performance level is faster than basic C programming
• C++ programming allows machine level workings to be performed.
• Like C programming language it has also crypto programming and security facilities along with error handling.

Cons:
• In compare to C programming it is tough to learn C++ as its modern language than C and require dedication.
• Its harder to working on C++ than C programming, but its output or results are better than C.[11]

2.4.3 Python

Pros:
• Working in python makes coding far more easier than before as it has amazing code readability quality, huge available free libraries to work on, already many built-in function and very easy syntax to remember.
• It allows to automate tasks and conduct malware analysis.
• In python programming language one can build their own tools of analyze, own hacking scripts. Programmers can also create their own hacking scripts to work on.
• Its does not require users to learn how a computer functions at the lowest level.

Cons:
• Unlike C/C++, Python is not low-level; therefore, it may took hard dedication to learn it.

2.4.4 Java

Pros:
• In Java it has a big advantages of having built-in API's in where one can implement their secure model by injection dependency of security API of java language.
• Again, another advantage of java programming can be use of SSL or secure socket layer which make the data package exchanging more secure.

Cons:
• Syntax is not as friendly as python. Takes lots of lines to execute a program.
• In Java language, concept of pointers is missing here as they wanted to prevent unauthorized user to corrupt one's system by use of pointers.

Comparison:

From the above pros and cons, [12] we can say that Python and C++ is best for working into security. Python will be used broadly in terms of internet connected areas and also in application. Additionally, C++ will be useful as there are so many resources in the security and networking area. That's why we will prioritize the C++ first and python in second as our language for the thesis work.

2.5 Database

Recently there are some databases that can be used in development projects like MySQL, MsSQL, NoSQL, Oracle and MongoDB. But among all of them we had to focus on which databases are more secure than usual. After researching we found that among all those databases MsSQL and Oracle are the most secure database.

Published security issues:

MsSQL and Oracle databases have published their own security issues among other databases. From those publications one can take extra care of those areas. For example, MsSQL has published about 36 security which they are now working on.[13]

Receive levels of security from security researcher:

Among the databases currently using, expert security researchers have given some levels of security. MsSQL and Oracle have got most of those levels.
From our research we find that MsSQL and Oracle have got the most recommendations. That's why we will prioritize MsSQL and Oracle one after another. Again, this 2 database has got most of the resources online which will be helpful in developing our ideas. Additionally, as security is one of the major concerns in our idea that's why we will divide the activity of databases among various devices of the network. That means all features of database insert, update, delete operation will be not done by only one device rather that it will be divided.[14]

2.6 Operating System

Kali Linux

In the area of ethical [15] hacking and entrance, Kali linux has been used and known for a while. It was first introduced by BackTrack and then Offensive Security. Additionally, kali linux is based on Debian which has a lot of entrance testing tools [16] to ensure security. Hundreds of hacking tools like forensic analysis, networking scanning, and crunch has been pre-installed in kali linux from various types of fields which are rolling release model. That means, every tool of this operating system will up to date always. Lastly, this operating system has a large community connection with given proper documentation where extensive amount of devices platform exist.

Chapter 3

Methodology

3.1 Proposed Idea

The target of our proposed model is to save the network and its database from outsiders. To meet this objective we set some layers of protection or methodology. First of all, every device in the network has to pass through the secret key. If the secret key does not match then the device information will be transferred to the thread alert. Additionally, we divide the workload of the database (insert, update, delete) into sensor, auv and node so that if one part gets affected the whole database does not come under attack. The figure 1 describes the general view of our proposed model.

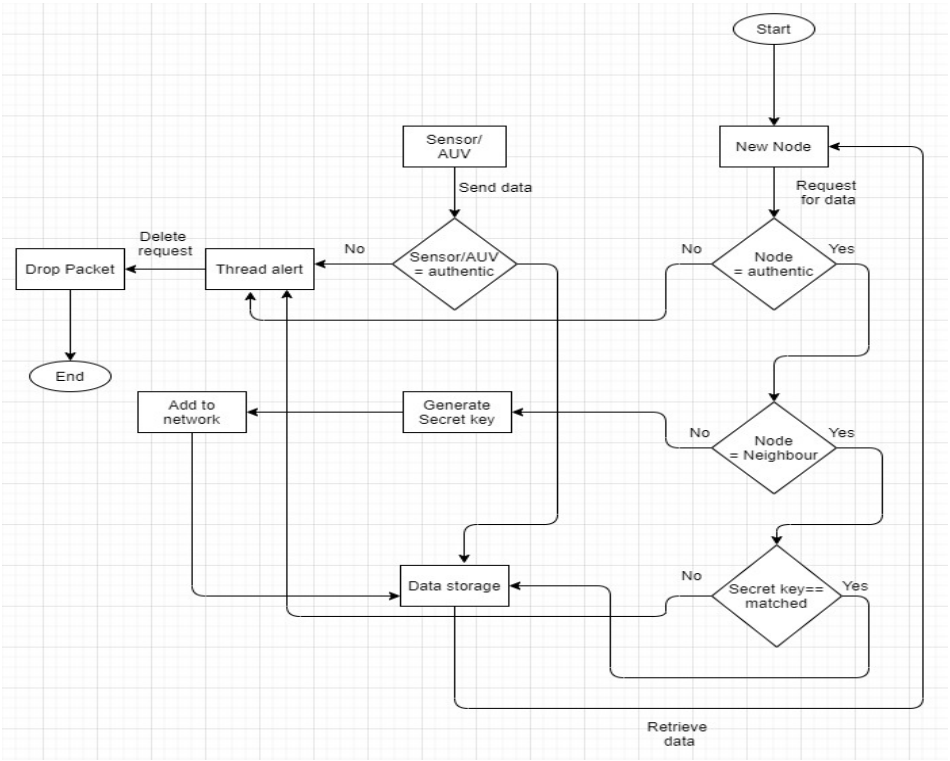


Figure 3.1: The flow chart of the proposed model

In the flow chart of our proposed model, we tried to visualize the whole idea where we divide it into two part. In node part, whenever a node will come to the network

it will face three different checking phase which are node authentic, node neighbor and secret key. In "node authentic" phase the system will alert e thread if the result comes out no where the request will go to next phase if the result comes out as yes. After that, in "node neighbor checking" phase a new secret will be generated adding to the network if the node is comes out as not neighbor and the request will go to next phase of checking which is "secret key match" if the result comes out as yes. In the last phase of checking for node, the node will retrieved the data if the result comes out as yes where a thread alert will initialized if the result comes out as no. In sensor AUV part, the device will face an authentication phase "Sensor/ AUV authentic" where the device will save data into network if the result comes out as yes and the system will initiate a thread alert like previously if the result comes out as no from the checking phase.

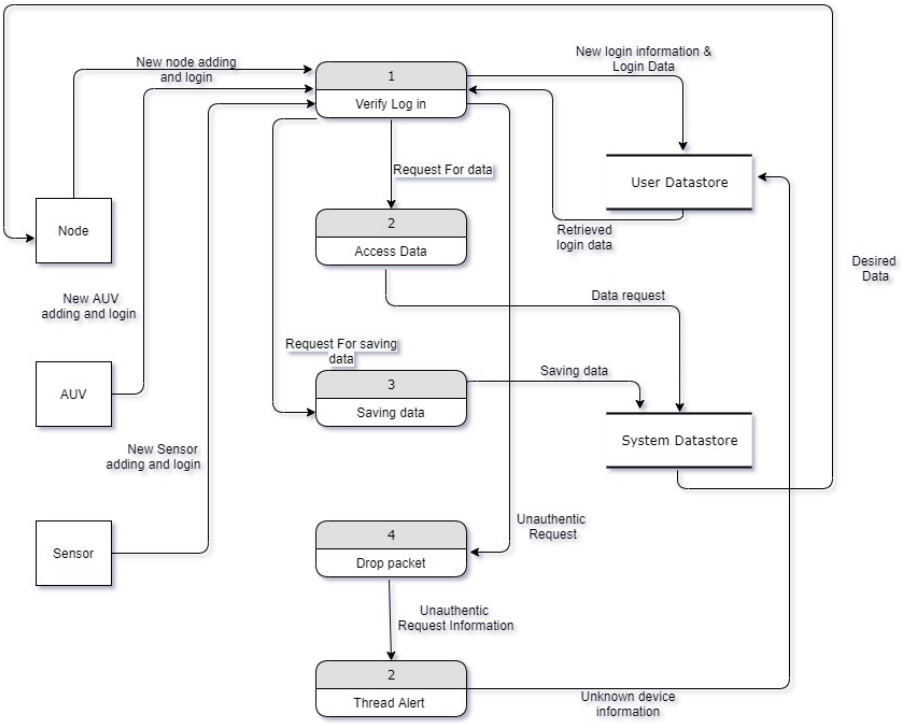


Figure 3.2: The data flow diagram of the proposed model

In data flow diagram of our project we can see how data is flowing from one process to another process. Main components of data flow diagram are rounded rectangle shaped process, rectangle shaped external entities, data stores and data flows of the system. At first, process produces a output line after getting a data input. In here, there are five process of the system which are verify log in, access data, saving data, drop packet, thread alert. Each of them receives some data line of input and then produces data line of output. For example, verify login data will receive log in new adding request from node, AUV, sensor. After getting this request verify log in process will verify this information from user data store by sending data to data store. After retrieving the verified login information from data store this process will entry authentic user and send their request to access data process where there will be another data path to the data store where data request will be go through verification. When verification is done, desired data will be send to the user. After

that, user can save data into the system data store through saving data process and there is a drop packet process which drop unauthentic request from the user. From the drop packet process data flow will move forward to the thread alert process where unauthentic request information is input and unknown device information is the output to the user data store of the system. This unauthentic device information is made for saving the history of unauthentic device or user.

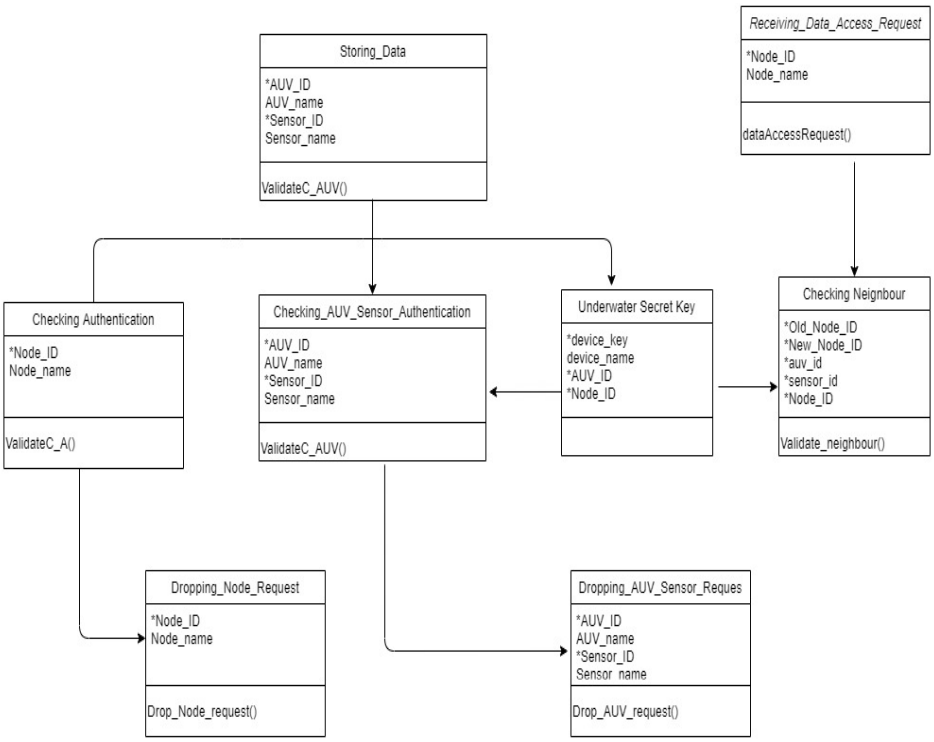


Figure 3.3: The class diagram of the proposed model

Underwater Secret key is our system's main class. The main four variables are Node Name, Node ID, AUV Name, AUV ID, Sensor Name and Sensor ID and among them ID variables are unique and also they are foreign key. This class has one method and it is main method. Inside of this method, there are six variables. Three variables for taking key from users for three different list and other three variables are for tracking the three keys of three different lists. Checking Authentication, Checking AUV Sensor Authentication and Checking Neighbor, this three classes are dependent on main class which is underwater secret key and vice versa. This three class and main class are working together to find out the authentication node, AUV and sensor. This class are using ID as foreign key. Dropping Node Request and Dropping AUV Sensor request are also depending on authentication class because by checking the authentication, our system decide which one's node, AUV or Sensor request to drop. Similarly Data access request and Storing Data classes are also depending on authentication classes. If the node or AUV are authentic then system will allow the node and AUV to perform databases task.

In the activity diagram, we can see that we have split the diagram for 2 types of device. For node, whenever a node will come to the network it will face an authentication check which will be done by checking against outsider list. In here, outsider list indicates the devices which have been tried to enter the network forcefully. Af-

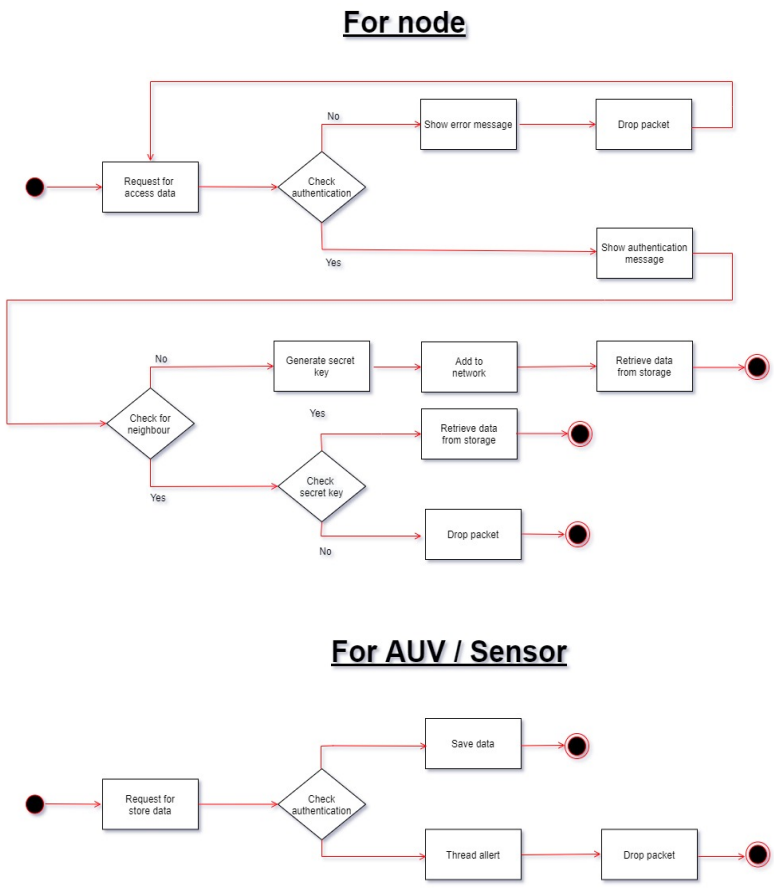


Figure 3.4: The activity diagram of the proposed model

ter that, if the node pass authentication phase successfully it will show successful message and it will show error message if the node is not authentic. Next, the data access request will be dropped if it is not authentic where in other side the requested node will face another checking where it will check if the node is neighbor or not. Additionally, this check will be done by checking against the list of neighbor's list of the network. Again, a secret key will generated and given to the node if the node is not neighbor or new to the network. After having secret key, the node will retrieve its desired data from the network's database and it will save its secret key for future communication.
On the other hand, like previously for AUV sensor type of device they will face authentication also and can save data into network's database if they proved authentic. In addition, this authentication will be done by checking against outsider list of AUV sensor.

In the use case diagram of our model we can see different phase of our proposed system where main components are indicated like actors, use case, system boundary and relations. In here, old node, new node, AUV and sensors are primary actor which are performing the main functions of the system. Additionally, database is the external actor as it is the external hardware device of our system. After that, use cases are showing in oval form which is basically the system functions or sequence of action. For example, check authentication, generate secret key, access data, store data, drop packet etc. Again, association relation are showing in between use case

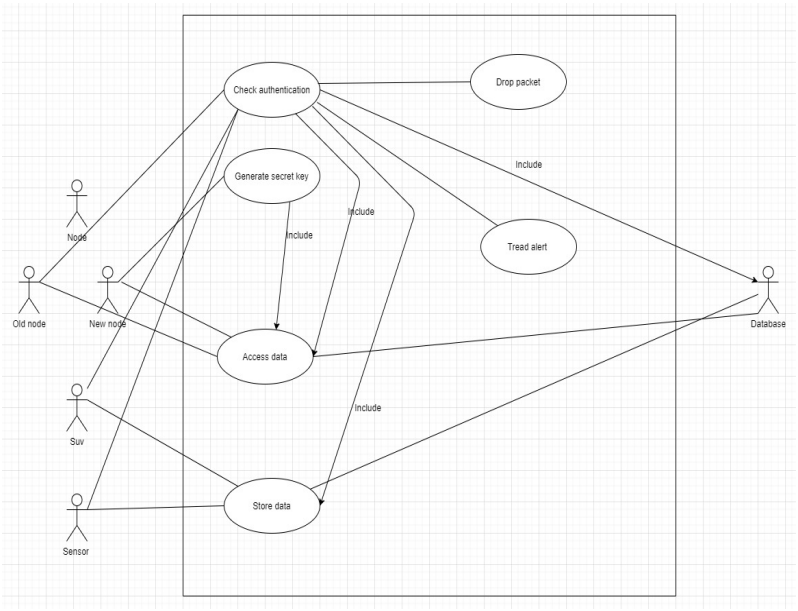


Figure 3.5: The use case diagram of the proposed model

and actors which is basically a straight line. Besides, some dependency types of relationship are also showing which is indicating the post condition. For example, in here for performing the store data one have to go through the authentication phase. So authentication is the post condition of storing the data.
This secure model process is focused on a secure network, maintaining a secure database and its operation. For that, this process model consists of four major stages:

1. Authentication
2. Secret key generate
3. Data input
4. Data retrieve

3.1.1 Authentication

The first stage of this proposed model is to ensure the authentication. Every device has to face the authentication before it comes under the network.

3.1.2 Secret Key Generate

The second stage is concern about the generating secret key if any new node comes to the network after authentication.

3.1.3 Data input

After verifying and facing the authentication all the sensors and AUV can input data to the database.

3.1.4 Data retrieve

Like previously after authentication and verification all the nodes can retrieve data from the database. Additionally, those nodes who are new to the network will be given a secret key generated by the network itself.

Chapter 4

Work and Analysis

4.1 Building Mechanism

We have used SOLID principle as our building mechanism as we have followed object oriented programming for our project. We used this principle because we wanted to keep our class single responsibility, shutting our classes-modules-functions for editing, reducing inheritance, making minimum interfaces and making dependency of abstractions among high level and low level instead of dependency of modules of high level and low level.

4.2 Classes

4.2.1 Checking Authentication

This is the system's first class. Checking Authentication class's purpose is to check a particular node is authentic to the system or not. Now a days security has become the major issue for every technology. Maintaining the surface security is easier than underwater security. Checking Authentication class has two libraries, among them one is built in C++ library and other is created by us because C++ doesn't have this particular library where Node of linked list has to be defined in our system's way. This class is approached solid methodology and it follows five principles of the solid methodology. Checking Authentication has two variables, among them one is Node Name and the other is Node ID. Node Name variable has string data type and Node ID has int data type. Checking Authentication class has one and only method as per solid methodology. The method name is validateChekingAuthentication. This method has void return type which means this method will not return any data. validateChekingAuthentication is receiving one argument from main method. Main method is sending one argument which is passing the data from Node library where we checking the node exists in the list or not. In the validateChekingAuthentication method system is checking that a particular node exists in the outsider list or not. If node exists in the outside list system will show "Node Already exists with key value in the Outsider list. So, it's Unauthentic Node" and if the node does not exist in the outsider list, then the system will show "Node does not exist in the outsider list. So, it's Authentic Node". That all from the Checking Authentication class.

4.2.2 Checking AUV Sensor Authentication

This is the system's second class. Checking AUV Sensor Authentication class's purpose is to check a particular AUV Sensor node is authentic to the system or not. Now a day's security has become the major issue for every technology. Maintaining the surface security is much easier than underwater security. Checking AUV Sensor Authentication class has three libraries, among them two libraries are built in C++ library and other is created by us because C++ doesn't have this particular library where Node of linked list has to be defined in our system's way. This class is approached solid methodology and it follows five principles of the solid methodology. Checking AUV Sensor Authentication has four variables, among them one is AUV Name and the others are AUV_ID, Sensor Name and Sensor ID. AUV Name and Sensor Name variable have string data type and AUV ID and Sensor ID have int data type. Checking AUV Sensor Authentication class has one and only method as per solid methodology. The method name is $validateC_AUV$. This method has int return type which means this method will return integer data. This integer data will help the system to determine that the node will perform task into the system's database or not. $validateC_AUV$ method is receiving one argument from main method. Main method is sending one argument which is passing the data from Node library where we checking the node exists in the list or not. In the $validateC_AUV$ method system is checking that a particular node exists in the AUV Sensor outsider list or not. If node exists in the AUV Sensor outsider list system will show "Node Already exists with key value in the AUV Sensor Outsider list. So, it's Unauthentic AUV Sensor" and in that case no task will be performed into the system's database. If the node does not exist in the AUV Sensor outsider list, then the system will show "Node does not exist in the outsider list. So, it's Authentic Node" and in that case the node can perform task into the system's database because the node is authentic node. That all from the Checking AUV Sensor Authentication class.

4.2.3 Checking Neighbor

This is the system's third class. Checking Neighbor class's purpose is to check a particular neighbor node is authentic to the system or not. Now a day's security has become the major issue for every technology. Maintaining the surface security is much easier than underwater security. Implementing technology in the underwater is very tough because there are lots of factor of the underwater environment to consider. Checking Neighbor class has two libraries, among them one library is built in C++ library and other is created by us because C++ doesn't have this particular library where Node of linked list has to be defined in our system's way. This class is approached solid methodology and it follows five principles of the solid methodology. Checking Neighbor class has three variables, among them one is Node Name and the others are $Node_ID$ and Secret key. Node Name variable has string data type and Node ID and Secret key have int data type. This secret key is our special key and it's the most important part of the system. The AES is helping for generating secret key. Checking Neighbor class has one and only method as per solid methodology. The method name is $validate_Neighbour$. This method has int return type which means this method will return integer data. This integer data will help the system to determine that the node will perform task into the system's database

or not. $Validate_Neighbour$ method is receiving one argument from main method. Main method is sending one argument which is passing the data from Node library where we checking the node exists in the list or not. In the $validate_Neighbour$ method system is checking that a particular node exists in the neighbor list or not. If node exists in the neighbor list system will show "Node exists with key value in the Neighbor list" and in that system will allow the neighbor node to perform any task into the system's database. If the node does not exist in the neighbor list, then the system will show "Node does not exist in the neighbor list." and in that case no task can be performed by the node into the system's database. That all from the Checking Neighbor class.

4.2.4 Dropping AUV Sensor Request

Our system has another class called Dropping AUV Sensor Request. By the name of the class, we can understand that Dropping AUV Sensor Request class's purpose is that this class will drop the request of AUV which are not authentic. Basically, this class will drop the request of node which wants to have access of the system's network or wants to join to the system's network. Dropping AUV Sensor Request class has four variables, two are string data type and other two are integer data type. Dropping AUV Sensor Request class has one and only method called Drop AUV Request() with an argument. Because of the solid methodology Dropping AUV Sensor Request class has one method. So, this class follows the five principles of solid methodology. Our system's main methods have the track of authentic and unauthentic AUV. From our system's main method this Drop AUV request method will receive an argument by which this will perform its task. If the AUV is not authentic, the system will show that "Since AUV is not authentic, the system has dropped the request of AUV.

4.2.5 Dropping Node Request

Our system has another class called Dropping Node Request. This class mechanism is also like previous class named Dropping AUV Sensor Request but Dropping Node Request is working with Node rather than handling AUV. By the name of the class, we can understand that Dropping Node Request class's purpose is that this class will drop the request of Node which are not authentic. Basically, this class will drop the request of node which wants to have access of the system's network or wants to join to the system's network. Dropping AUV Sensor Request class has two variables, one string data type and other one is integer data type. Dropping Node Request class has one and only method called Drop Node Request() with an argument. Because of the solid methodology Dropping Node Request class has one and only method. So, this class follows the five principles of solid methodology. Our system's main methods have the track of authentic and unauthentic Node. From our system's main method this Drop node request method will receive an argument by which this will perform its task. If the Node is not authentic, the system will show that "Since Node is not authentic, the system has dropped the request of Node.

4.2.6 Receiving Data Access Request

Receiving Data Access Request is another class for our system. This class also follows solid methodology and its five principles as similar as the other classes follow solid methodology. This class has one and only method called dataAccessRequest() with an argument and its return type is void which means this method is not returning any value. Our system's main method of UnderwaterSecreatKey class has the track of authentic and unauthentic node. Form our system's main method dataAccessRequest() method will receive an argument and with that argument method will check if the AUV is authentic then system will allow the AUV to access system network and its database. Receiving data access request method is working with AUV. So system will the access to the AUV if the AUV is authentic. Finally, AUV perform its task and perform task on system's database.

4.2.7 Storing Data

This is our system's another class where the tasks of database are performed. Storing data is following solid methodology and its five principle. This class has one method like previous classes because every classed of our system is following solid methodology. Function of this class is receiving an argument from the main function of the main class and with that function is verifying the data. Once the node get verified, system will allow the node for retrieve operations

4.2.8 Underwater Secret Key

This is our system final class where our system's main method is located. In this class all the task of the system will be performed. This class is combined with thirteen different libraries. Five libraries are created manually as per our system's needs which means those are not built in by C++ library and other four are system's different class files and the rest of the libraries are built in the C ++ library. This class has one and only method named main method as per the solid methodology. Our whole system follows solid methodology and its five principle. Here in the main our system has used linked list instead of array for creating three different node list. Linked list instead of array because array data structure has its own drawbacks over linked list. The main disadvantages of array are fixed capacity and shifting elements while inserting or removing elements. Our system's needs very simple application and huge amount of space. To overcome those array data structure problem, our system is using linked list where system can easily insert elements anywhere of the list and no need to define list capacity as it holds space dynamically in the computer's memory (RAM). There is another reason for using linked list to implement the key idea. From online resource we have found that key idea will be very handful for our system and it will serve the uniqueness idea to the secret key. This key is using to give uniqueness to each and every node. Since, there is global pandemic situation, we had to developed a software approached system. That's why our system is using predefined variable. Our system is assuming that three lists are receiving through sensors, AUV and its node. Hence, we have created three different linked list to differentiate and track the nodes. Checking Authentication, Checking AUV Sensor Authentication and Checking Neighbor are three different linked list. How the linked list created and how they performing are described below.

First, Outsider list will be created with a data and a unique key which is defined as Node ID. then system will ask for a key (Node) input from a user. Since it is a software-based system and for global pandemic we didn't include hardware implementation. For that reason, it will take input from the user at this moment. After taking the key (Node) form the user, system will check the key (Node) against Outsider list key (Node) value which have been defined previously. If key (Node) exists in the Outsider list, system will show key (Node) is not authentic which means this is an outsider node and unauthorized node and it is danger for the system. If key (Node) doesn't exist in the Outsider list, system will show key (Node) is authentic which means the key (Node) is not an outsider, it has already been to our system network.

Secondly, AUV Sensor outsider list will be created with a data and a unique key in the same way we created outsider list previously. This AUV Sensor outsider linked list is the linked list. Then, system will ask for a key (Node) from a user to check that key (Node) exists or not in the AUV Sensor outsider list. Since it is a software-based system and for global pandemic we didn't include hardware implementation. For that reason, it will take input from the user at this moment. Otherwise, it would have taken the input from the sensor automatically. After taking the input, if key (Node) exists in the AUV Sensor outsider list, the system will show key (Node) is not authentic which means the node is unauthorized node and it is danger to the system and also the system will not allow any task to be performed into the database. If key (Node) doesn't exist in the AUV outsider list, the system will show key (Node) is authentic which means the node is safe and authorized node and the system will allow any to be performed into the system's database.

Thirdly, Neighbor list will be created with a data and a unique key in the same way we have created other two linked list for the system. Here the key is an idea into linked list to give each node to be uniqueness. System will ask for a key (Node) from the user to check that key (Node) is exist or not in the neighbor list. Since it is a software-based system and for global pandemic we didn't include hardware implementation. For that reason, it will take input from the user at this moment. If key (Node) exists in the Neighbor list, the system will show that key (Node) exists in the neighbor list which refers that the node is belong to system's network and the system will allow the key (Node) to retrieve data from system's database. If key (Node) doesn't exist in the neighbor list, system will show that the key (Node) doesn't exist in the Neighbor list which refers that the node has never been to the system's network and the system will deny the key (Node) to perform any task in the system's database.

At the end. We have included some operation of linked list for example printing the list, appending a node in the list and deleting a node from the list etc. These operations are taking place with the help of node library which we have created according to our system necessity.

4.3 Header Files

4.3.1 Node

This is our system's first library file which was created by us manually for linked list implementation. We know that linked list works with node data type. Node data type is not built-in data type of programming language. For that reason, we had to build node header file as per our system. We also included another new idea which is key. Key serves no special purpose but this idea came very useful for our system. Key can be used in liked list and it is just another integer type data. We wanted to implement uniqueness to each node and it had possible for key idea. While researching online about liked list, we have gotten this idea. This key idea will give our system to another level of security. By concatenating key and data, our system is generating secret key by using advanced encryption standard (AES). There are two class and two constructors for each class. Our system is using singly linked list. We are also implementing dynamic memory allocation by using pointer. There are seven methods for seven different application of singly linked list.
A node exists or not in singly linked list, for that we are using nodeExists method. From the main method of UnderwaterSecretKey class, several methods are passing an argument with a key value and this nodeExists method checking the node exists or not in the particular singly linked list with the key value. Then we have appendNode method. Basically, this method is joining a new node after end of the linked list with the new key value. Then we have prependNode method and this method is adding new node in the beginning of linked list with new key value. Then we have insertNodeAfter method and this is adding new node with new key after a particular node. After that, our node header files another method called deleteNodeByKey and this method is performing a delete task which means this method is deleting a particular node by particular key. Then we have another method called updateNodeByKey and this method also working as same way the previous method is working but it is performing update task. Finally we have last method of node header file is print method to print our all three different linked list.

4.3.2 Generating Secret Key

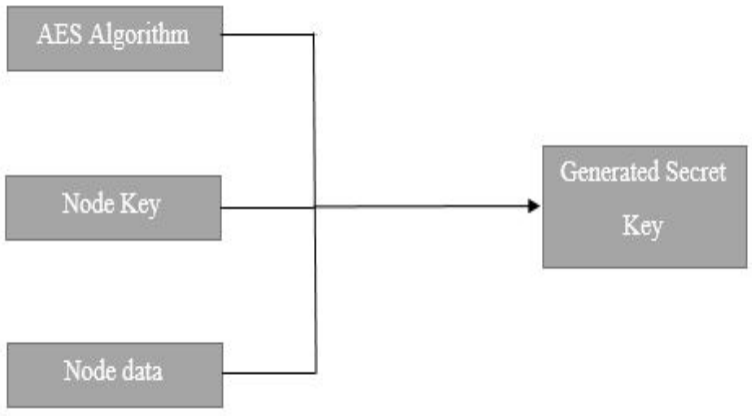


Figure 4.1: Secret key generation diagram of proposed model

Generating the secret key is the most significant part of our proposed model where we have used 3 types of data for building the secret key which are secret key generated from typical AES Algorithm, key of our predefined node and the data of our predefined node. We are mixing these 3 data for generating a very strong key than the typical secret key available right now.

4.4 Database Query

4.4.1 SQLInsert

This is our system's another customized header files for establishing connection with system's database and performing insert operation. First of all, we had to create systems with its data which our system needed. Our database table name is testdb and we have set user id myID and we have also set password to 1234 and its server local host is 1234. In this header files, we have included several libraries. In this file, we have also included several functions for different purposes. Our first function of this file is showSQLError() and its type is void type which means it will not return any value. This will show all possible while executing SQL command in C++. Another function is SQLGetDiagRec to perform a diagnosis on real time to see if there is any error. There might be an error while setting the variable because of using SQLWCHAR type. This error can be solved by setting the character set in general setting from Use Unicode Character Set to Use Multi-Byte Character Set. After that, we need to make sure that our configuration to active(Release) and platform to x64. Then we have declared the callInsert function where system will perform the insert operation. After that, we created do cycle which will run only once. Using SQLAllocHandle for allocating the environment. Using SQLSetEnvAttr for setting the environment attributes which will govern the environment. Using SQLSerConnectAttr for setting the connection attributes which will establish the connection. In this function, time out has set for the connection. Using SQLDriverConnect function for actually connecting the system with the database to actuate to the SQL server and this function has a lot of inputs for avoiding a prompt for the user. We need to specify the type, port and the login for the database. This is how our system is performing insert operation in the system's databse.

4.4.2 SQLRetrive

This is our system's another customized header files for establishing connection with system's database and performing retrieve operation. First of all, we had to create systems with its data which our system needed. Our database table name is testdb and we have set user id myID and we have also set password to 1234 and its server local host is 1234. In this header files, we have included several libraries. In this file, we have also included several functions for different purposes. Our first function of this file is showSQLError() and its type is void type which means it will not return any value. This will show all possible while executing SQL command in C++. Another function is SQLGetDiagRec to perform a diagnosis on real time to see if there is any error. There might be an error while setting the variable because of using SQLWCHAR type. This error can be solved by setting the character set in general setting from Use Unicode Character Set to Use Multi-Byte Character

Set. After that, we need to make sure that our configuration to active(Release) and platform to x64. Then we have declared the callRetrieve function where system will perform the retrieve operation. After that, we created do cycle which will run only once. Using SQLAllocHandle for allocating the environment. Using SQLSetEnvAttr for setting the environment attributes which will govern the environment. Using SQLSerConnectAttr for setting the connection attributes which will establish the connection. In this function, time out has set for the connection. Using SQLDriverConnect function for actually connecting the system with the database to actuate to the SQL server and this function has a lot of inputs for avoiding a prompt for the user. We need to specify the type, port and the login for the database. This is how our system is performing retrieve operation in the system's database.

Chapter 5

Result Analysis

Secret key from the model we have been designed have the total 166 pair of words which is far bigger than traditional secret key that generated from AES algorithm. One example of generated secret key from our model is:
6B 0A 3E F2 2E 6E BF C1 DF C8 20 E6 35 09 67 49 6B 0A 3E F2 2E 6E BF C1 DF C8 20 E6 35 09 67 49 6B 0A 3E F2 2E 6E BF C1 DF C8 20 E6 35 09 67 49 6B 0A 3E F2 2E 6E BF C1 DF C8 20 E6 35 09 67 49 6B 0A 3E F2 2E 6E BF C1 DF C8 20 E6 35 09 67 49 6B 0A 3E F2 2E 6E BF C1 DF C8 20 E6 35 09 67 49 6B 0A 3E F2 2E 6E BF C1 DF C8 20 E6 35 09 67 49 6B 0A 3E F2 2E 6E BF C1 DF C8 20 E6 35 09 67 49 6B 0A 3E F2 2E 6E BF C1 DF C8 20 E6 35 09 67 49 6B 0A 3E F2 2E 6E BF C1 DF C8 20 E6 35 09 67 49 6B 0A 3E F2 2E 6E
Again, it takes less than 4 seconds to generate the key which is very low amount of time compared to this length of key.

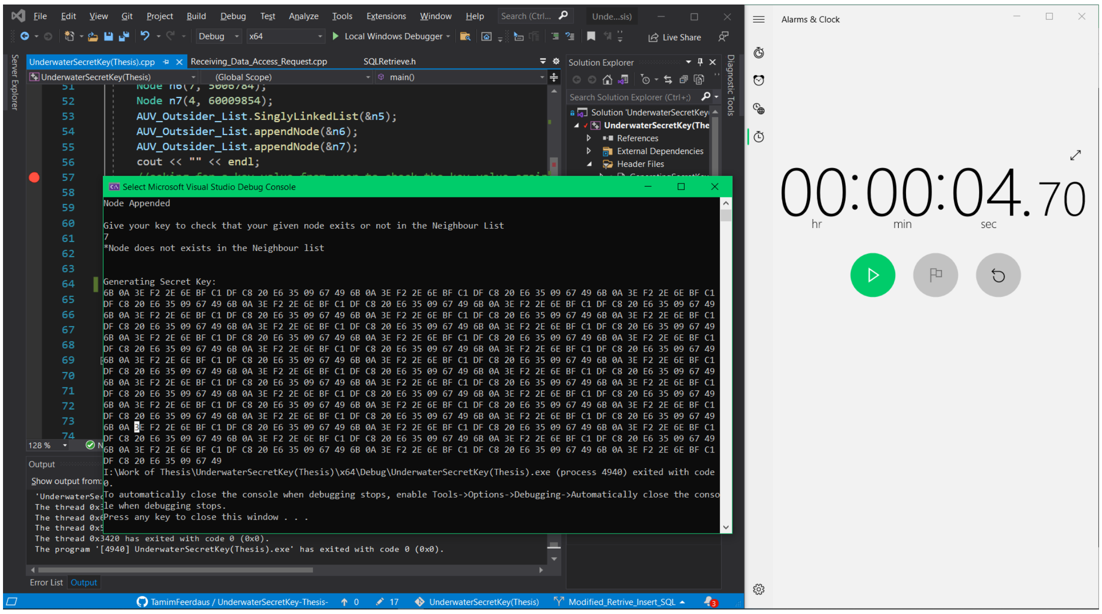


Figure 5.1: Screenshot of secret key with time

After that, we have tried to break the secret key by kali linux operating system. Author Noura Alesia in her paper [17] called “A comparison of the 3DES and AES encryption standards” state that it will take (5×1021) years to break the secret key that generated from AES algorithm when the input keys will be 50 billion per second [10]. For that reason, it will take the double period of breaking time than 128

length of secret key to break 256 length of secret key. Additionally, our generated key is bigger than AES algorithm's key and that's why it looks impossible when any outsider will take attempt to break it. However, for visualizing practically we have been used crunch method of kali linux brute force algorithm for almost one third of a day to break our generated secret key for our designed system, but it could not break it.

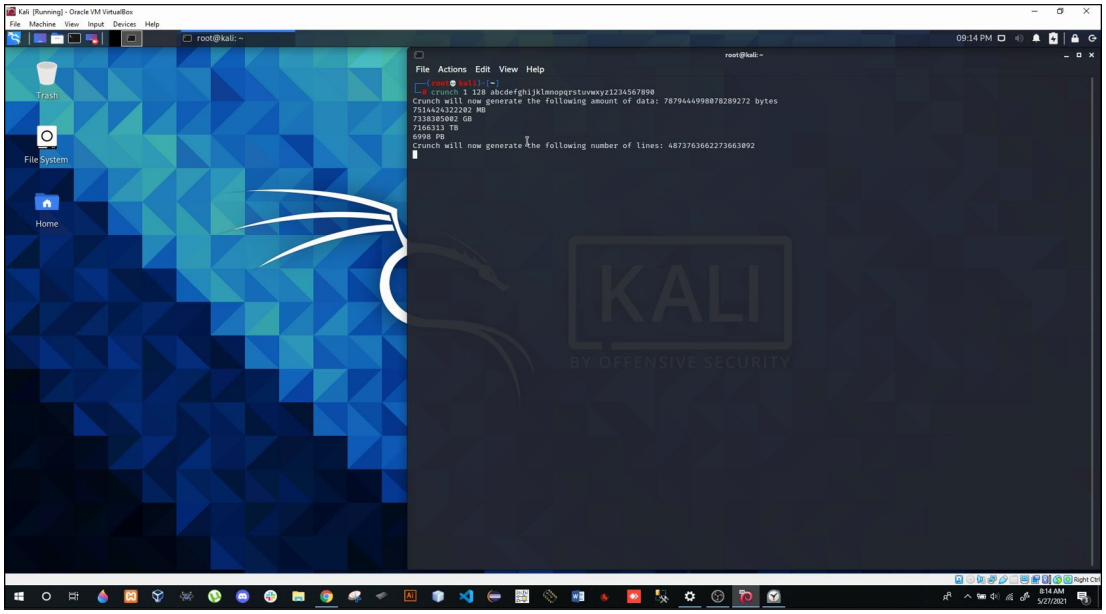


Figure 5.2: Screenshot of crunch algorithm running in kali linux

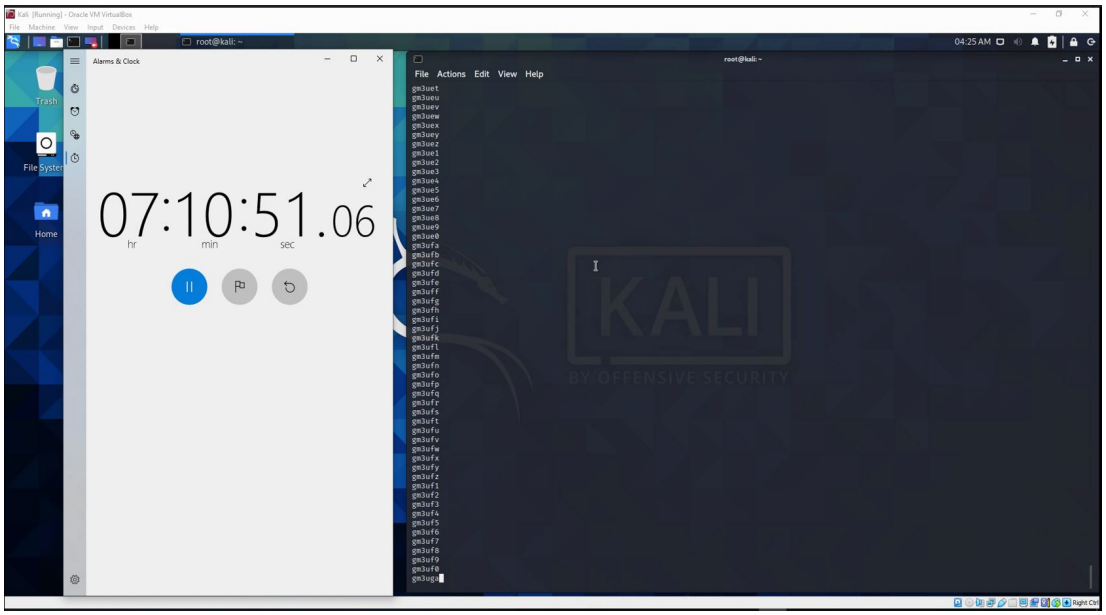


Figure 5.3: Screenshot of crunch algorithm running in kali linux for seven hours

Chapter 6

Limitations and Future Analysis

Currently we are implementing our idea in software presentation due to world pandemic situation. Besides, we are planning to implement our proposed model on hardware where we will use the real sensors, AUV and node under water as soon as the recent pandemic situation get normal.
Next, for our future updating works, we will use more combination of character like (, @,,*) to make our generated secret key more stronger.
Again, we will make more smooth connection in our database by connecting the tables to run the system in more convenient way.
After that, we will create more database admin with limited work to handle the database and the system properly.

Chapter 7

Pseudo Code

Checking Authentication Class
Two variables for Nodes' Name and ID

```
validateChecking_Authentication (node type data of nodes key
value will be passed from main method)
    countOut variable for tracking the process of main method

    if the key value is not null then it will show that node
    is not authentic

    else it will show authentic
end
```

Checking AUV Sensor Authentication Class
two variables for AUV Name and ID
two variables for Sensor Name and ID

```
validateC_AUV (node type data of AUVs and Sensors key value
will be passed from main method)
    countAUV  variable for tracking the process of main method

    if the key value is not null then it will show that AUV
    is not authentic and its outsider

    else it will show that it is authentic and it is not an
    outsider
end
```

Checking Neighbor Class
Two variables for Nodes' Name and ID
One variable for Secret key

```
Validate_Neighbor(node type data of nodes key value will be
passed from main method)
    countOut  variable for tracking the process of main method

    if the key value is not null then it will show that node
    is authentic and will also allow to access database
```

```
    else it will show that it is not authentic cause it is
    not exist in the neighbor list
end
```

Dropping AUV Sensor Request Class
Two variables for AUV Name and ID
Two variables for Sensor Name and ID

```
Drop_AUV_Request(integer type data of count will be passed
from main method)
        //here count variable is tracking the process of
        main method

        If count is zero then it will show request of AUV
        has been dropped
end
```

Dropping Node Request Class
Two variables for Nodes Name and Node ID

```
Drop_Node_Request(integer type data of count will be passed
from the main method)
        //here count variable is tracking the process of
        main method

        If count is one then it will show your request has
        been dropped
```

Receiving Data Access Request Class
Two variables for Node Name and Node ID

```
dataAccessRequest(integer type data of count will be passed
from the main method)
        //here count variable is tracking the process of
        main method

        If count is one then it will show data access request
        of node has been granted and node will perform
        operations of database
end
```

Storing Data Class
Two variables for AUV Name and AUV ID
Two variables for Sensors Name and Sensor ID

```
Data_Store( integer type data of count will be passed
from the main method )
        //here count variable is tracking the process of
        main method
        If count is one then it will allow the AUV to store
        data into database
end
```

Underwater Secret Key Class

// main class of the project

// this class will multiple libraries which is not built-in libraries which means it's fully customized

Object of Outsider Linked List

Here predefined outsider singly linked list will be created for Node

Ask for a key form the user to check that particular key of AUV is in the list or not

Key will be passed to Node library which will return the address of that particular key and then it will pass the address to the checking authentication class to check that node's existence

After that count will track the process and it will pass to the checking AUV authentication method

Then system will call drop node request to drop the request by checking previously that node exist or not.

Object of AUV Outsider Singly Linked List

Here predefined AUV outsider Singly linked list will be created for AUV

Ask for a key form the user to check that particular key of AUV is in the list or not

Key will be passed to Node library which will return the address of that particular key of AUV and then it will pass the address to the checking AUV authentication class to check that AUV's existence

After that count will track the process and it will pass to the checking AUV authentication method

Then system will call drop AUV request to drop the request by checking previously that AUV exist or not

Then system will call store data method to from storing data class for inserting data into database if AUV does not exist in the AUV outsider list

Object of Neighbor Singly Linked List

Here predefined Neighbor Singly linked list will be created for node

Ask for a key form the user to check that particular key of node is in the list or not Key will be passed to Node library which will return the address of that particular key of node and then it will pass the address to the checking Neighbor class to check that Node's existence After that count will track the process and it will pass to the checking neighbor method

Then system will call data access request to accept the request by checking previously that Node exist or not Then system will call retrieve library for retrieving data from database

At the end of this class secret key will be generated by the library System will send all of its node's concatenated data and key to the library for generating secret key

Chapter 8

Conclusion

In conclusion, we would like to say that network security solutions are a great way to secure our information and data. There will always be a gap in the solutions but our goal to minimize those issues. By implementing our proposed idea will bring another level of security in the underwater network. This will be another level of security and it will minimize the gap in the underwater network security. The secret key will be the main key to provide or maintain the underwater network security. Thus our research represents an attempt to fill this gap by adding another layer of protection to the already existing layers or the layers that are being developed for underwater network security systems. By doing so we can keep a secure underwater environment as well as surface environment.

Bibliography

[1] I. G. A. Awan, Shah, "Underwater wireless sensor networks: A review of recent issues and challenges," *Wireless Communication and Mobile Computing*, vol. 2019, no. 1, pp. 167–173, 2019.

[2] D. Pompili and I. F. Akyildiz, "Overview of networking protocols for underwater wireless communications," *IEEE Communications magazine*, vol. 47, no. 1, pp. 97–102, 2009.

[3] M. C. Domingo, "Securing underwater wireless communication networks," *IEEE Wireless Communications*, vol. 18, no. 1, pp. 22–28, 2011.

[4] A. Davis and H. Chang, "Underwater wireless sensor networks," pp. 1–5, 2012.

[5] P. Nicopolitidis, G. I. Papadimitriou, and A. S. Pomportsis, "Adaptive data broadcasting in underwater wireless networks," *IEEE Journal of Oceanic Engineering*, vol. 35, no. 3, pp. 623–634, 2010.

[6] V. Rodoplu and M. K. Park, "An energy-efficient mac protocol for underwater wireless acoustic networks," pp. 1198–1203, 2005.

[7] Z. Li, N. Yao, and Q. Gao, "Relative distance based forwarding protocol for underwater wireless networks," *International Journal of Distributed Sensor Networks*, vol. 10, no. 2, p. 173089, 2014.

[8] K. Chen, M. Ma, E. Cheng, F. Yuan, and W. Su, "A survey on mac protocols for underwater wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1433–1447, 2014.

[9] Y. Luo, L. Pu, Z. Peng, and Z. Shi, "Rss-based secret key generation in underwater acoustic networks: advantages, challenges, and performance improvements," *IEEE Communications Magazine*, vol. 54, no. 2, pp. 32–38, 2016.

[10] G. Singh, "A study of encryption algorithms (rsa, des, 3des and aes) for information security," *International Journal of Computer Applications*, vol. 67, no. 19, 2013.

[11] M. J. Garbade, "5 best programming languages to learn for cyber security," vol. 2021, no. 1, p. 1, 2021.

[12] J. Robert C, "The best programming languages to learn for cybersecurity - springboard blog," vol. 1, no. 1, pp. 1–2, 2021.

[13] A. David, "The database hacker's handbook: Defending database servers," *https://www.oreilly.com/library/view/the-database-hackers/9780764578014/ch001-sec001.html*, vol. 1, no. 2, pp. 1–2, 2021.

[14] K. Miller, "Database security tools," *https://www.trustradius.com/database-security*, vol. 1, no. 1, p. 1, 2021.

[15] M. Okoi, "Top 15 best security-centric linux distributions of 2020," vol. 1, no. 1, p. 1, 2021.

[16] M. Tanjim, "Best linux distributions for hacking and penetration testing," vol. 1, no. 1, p. 1, 2021.

[17] N. Aleisa, "A comparison of the 3des and aes encryption standards," *International Journal of Security and Its Applications*, vol. 9, no. 7, pp. 241–246, 2015.

Appendix

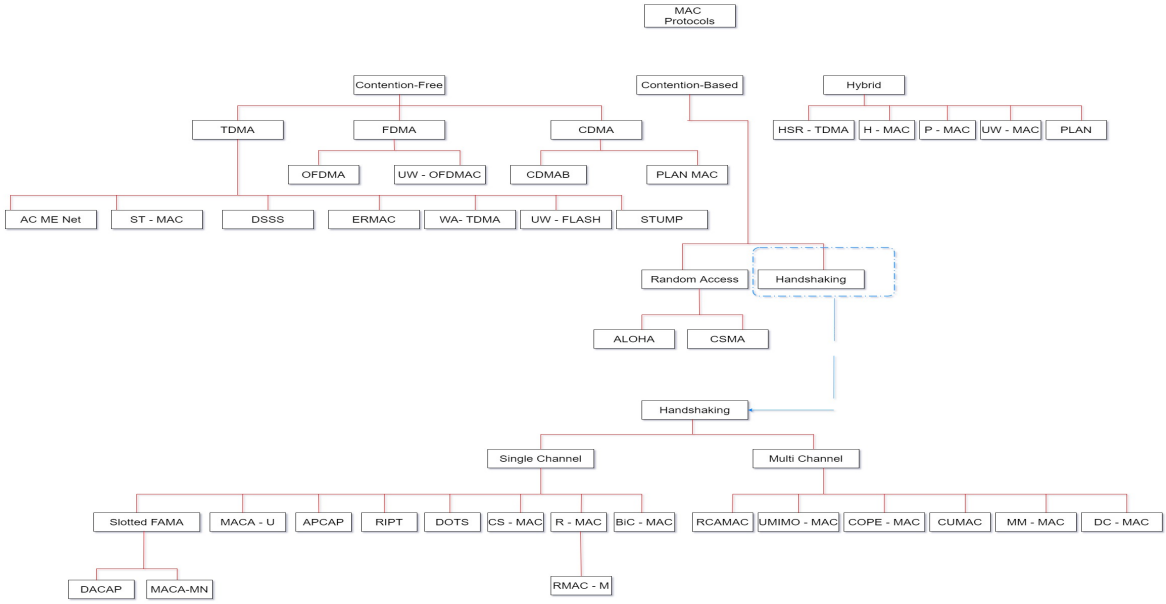


Figure 8.1: The classification of MAC Protocols for UWSNs[8]

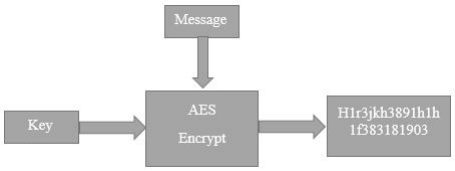


Figure 8.2: Encryption diagram of AES

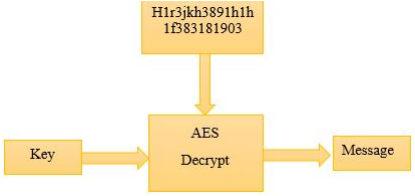


Figure 8.3: Decryption diagram of AES

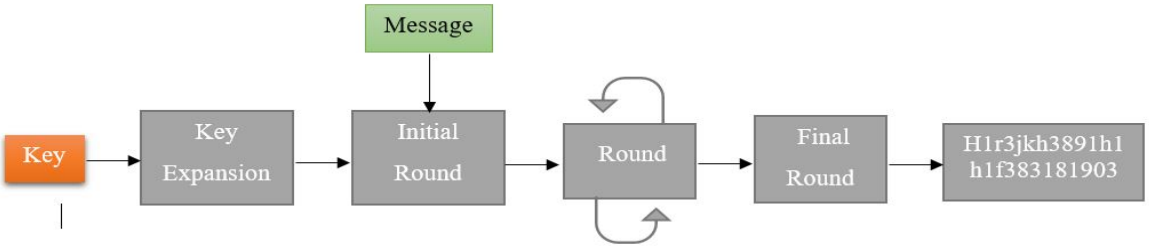


Figure 8.4: Full process of AES secret key generation