**An Efficient Privacy Preserving Message**

**Authentication Scheme for Internet of Things**

A Project Report submitted in partial fulfillment of the degree of the   
 **Bachelor of Technology** in **Computer Science and Engineering**

By

GATTLA NAVEENA 20C45A0508

CHIRRA SRAVANALAXMI 20C45A0510

DONTHULA RAVALI 20C45A0511

Under the Guidance of

**Mr. G. Pandukrishna**

(Assistant Professor)



**Department of Computer Science and Engineering**

**Jayamukhi institute of technological Sciences**

**Narsampet, warangal-506 332**

**(UGC AUTONOMUS, Affiliated to JNTUH, Accredited by NAAC ‘A’ Grade)**

**2023**

**Jayamukhi institute of technological Sciences**

**Narsampet, warangal – 506 332**

**(UGC A, Affiliated to JNTUH, Accredited by NAAC ‘A’ Grade)**

**CERTIFICATE**



This is to certify that the Project Report entitled **“AN EFFICIENT PRIVACY PRESERVING MESSAGE AUTHENTICATION SCHEME FOR INTERNET OF THINGS”** is a bonafide work of the students **G.NAVEENA(20C45A0508), Ch.SRAVANALAXMI(20C45A0510), D.RAVALI(20C45A0511)** submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering during the academic year 2022-23.

**Guide Head of the Department**

**Principal**

# **ACKNOWLEDGEMENT**

We express our immense gratitude and sincere thanks to **Dr. V. JANAKI**, Principal, Jayamukhi Institute of Technological Sciences for her constant motivation and support.

We are Extremely thankful to **Dr. P. SRINIVAS RAO**, Head of the Department of Computer Science and Engineering for the Support and encouragement according to carry out this project work.

We would especially like to express out extreme gratitude and sincere thanks to our guide **Mr. G. PANDUKRISHNA,** Asst. Prof, Department of Computer Science and Engineering for his enthusiastic and innovative guidance and support.

We are also thankyou to our management for providing all the facilities for completing the project

THANK YOU

G.NAVEENA -20C45A0508

CH.SRAVANALAXMI-20C45A0510

D.RAVALI-20C45A0511

**ABSTRACT**

As an essential element of the next generation Internet, Internet of Things (IoT) has been undergoing an extensive development in recent years. In addition to the enhancement of people’s daily lives, IoT devices also generate/gather a massive amount of data that could be utilized by machine learning and big data analytics for different applications. Due to the machine-to-machine (M2M) communication nature of IoT, data security and privacy are crucial issues that must be addressed to prevent different cyber attacks (e.g., impersonation and data pollution/poisoning attacks). Nevertheless, due to the constrained computation power and the diversity of IoT devices, it is a challenging problem to develop lightweight and versatile IoT security solutions. In this paper, we propose an efficient, secure, and privacy-preserving message authentication scheme for IoT.

Our scheme supports IoT devices with different cryptographic configurations and allows offline/online computation, making it more versatile and efficient than the previous solutions.

**CONTENTS**

**S.NO NAME OF THE TOPIC PAGE NO**

1. **INTRODUCTION 7**
2. **LITERATURE REVIEW 9**
3. **REQUIREMENTS SPECIFICATIONS 13**
   1. Existing System
   2. Disadvantages of Existing System
   3. Proposed System
   4. Advantages of Proposed System
   5. Software and Hardware requirements
4. **SYSTEM STUDY 16** 
   1. Feasibility Study
   2. Economical Feasibility
   3. Technical Feasibility
   4. Social Feasibility
5. **SYSTEM DESIGN 17**
   1. System Architecture
   2. Data Flow Diagram
   3. Flow Chat
   4. UseCase Diagram
   5. Class Diagram
   6. Sequence Diagram
6. **IMPLEMENTATION 27**
   1. Modules and their Description
7. **SOFTWARE ENVIRONMENT 28**
   1. Java Technology
   2. Java Programming
   3. Java Platform
   4. ODBC
   5. JDBC
   6. Networking
   7. JFree Chart
   8. J2ME
8. **SYSTEM TESTING 42**
   1. Testing Methodologies
   2. User Training
   3. Maintenance
   4. Testing Strategies
9. **OUTPUT SCREENS 46**
10. **CONCLUSION 51**
11. **REFERENCE 52**
12. **INTRODUCTION**

It allows devices to simplify the retrieval as well as the exchange of data without human involvement in various applications [1] and has a considerable position in the growth of information technology after the computer science and the Internet. IoT brings a pervasive digital appearance by engaging society and industries, and enables a series of interactions between human to human, human to thing, and more importantly, thing to thing. The development of IoT has led to enormous applications, such as smart home systems (SHSs) [2], intelligent transportation systems [3][4], machine learning and big data [5], etc. The machine-to-machine (M2M) [6] communication among massive numbers of IoT devices will dominate future communication network traffic. The integrity and authenticity of the massive amount of data collected and transmitted by the IoT devices are crucial in some applications such as machine learning and big data analytics. Maliciously injected or modified data can cause biased or wrong decision making and prediction. Therefore, in order to ensure the correctness and accuracy of machine learning and big data analysis, the integrity and authenticity of the collected data must be retained [7].

There are two approaches to achieve secure message delivery in IoT: the symmetric-key based approach, and the publickey based approach. The symmetric-key approach incurs less computation overhead compared with the public-key approach since symmetric-key operations are much more efficient than their public-key counterparts. However, key management is a major issue for symmetric-key based approach in a large scale heterogeneous IoT network. Also, if the message is only authenticated using a shared key between the sender and the receiver, the intermediate forwarding nodes in the IoT network cannot verify the integrity of the message. If the message has been altered or damaged during transmission, then the problem can only be discovered by the receiver. On the other hand, public-key based approach can solve these problems since anyone can use the public key to verify the integrity and authenticity of a message. However, public-key operations are very computation intensive, and privacy is another concern for public-key based approach since the authentication token is publicly verifiable using the sender’s public key. It is worth noting that the privacy of a data source is also important in some situations, e.g., when a wearable device is attached to a human. If the attacker can identify the sources of the data streams, then they could also cut off a data stream (e.g., via a Denial-of-Service attack) and eventually affect the accuracy of the decision or prediction produced by machine learning.

In order to address the above problems in IoT and M2M communications, a secure, efficient and privacy-preserving message authentication scheme that can support hop-by-hop verification is desirable. In [8], Li et al. proposed a novel source anonymous message authentication (SAMA) scheme which could be used for such a purpose. Their scheme was believed to achieve message authentication and message source privacy with a lower cost than the previous approaches.

**Introduction to Wireless Sensor Networks**

With the popularity of laptops, cell phones, PDAs, GPS devices, RFID, and intelligent electronics in the post-PC era, computing devices have become cheaper, more mobile, more distributed, and more pervasive in daily life. It is now possible to construct, from commercial off the shelf (COTS) components, a wallet size embedded system with the equivalent capability of a 90's PC. Such embedded systems can be supported with scaled down Windows or Linux operating systems. From this perspective, the emergence of wireless sensor networks (WSNs) is essentially the latest trend of Moore's Law toward the miniaturization and ubiquity of computing devices.

Typically, a wireless sensor node (or simply sensor node) consists of sensing, computing, communication, actuation, and power components. These components are integrated on a single or multiple boards, and packaged in a few cubic inches. With state-of-the-art, low-power circuit and networking technologies, a sensor node powered by 2 AA batteries can last for up to three years with a 1% low duty cycle working mode. A WSN usually consists of tens to thousands of such nodes that communicate through wireless channels for information sharing and cooperative processing. WSNs can be deployed on a global scale for environmental monitoring and habitat study, over a battle field for military surveillance and reconnaissance, in emergent environments for search and rescue, in factories for condition based maintenance, in buildings for infrastructure health monitoring, in homes to realize smart homes, or even in bodies for patient monitoring.

After the initial deployment (typically ad hoc), sensor nodes are responsible for self-organizing an appropriate network infrastructure, often with multi-hop connections between sensor nodes. The onboard sensors then start collecting acoustic, seismic, infrared or magnetic information about the environment, using either continuous or event driven working modes. Location and positioning information can also be obtained through the global positioning system (GPS) or local positioning algorithms. This information can be gathered from across the network and appropriately processed to construct a global view of the monitoring phenomena or objects. The basic philosophy behind WSNs is that, while the capability of each individual sensor node is limited, the aggregate power of the entire network is sufficient for the required mission.

**2. LITERATURE REVIEW**

**L[1]: Internet of things in industries**

**Authors:** L. Da Xu, W. He, and S. Li

**Description:** Internet of Things (IoT) has provided a promising opportunity to build powerful industrial systems and applications by leveraging the growing ubiquity of radio-frequency identification (RFID), and wireless, mobile, and sensor devices. A wide range of industrial IoT applications have been developed and deployed in recent years. In an effort to understand the development of IoT in industries, this paper reviews the current research of IoT. key enabling technologies, major IoT applications in industries, and identifies research trends and challenges. A main contribution of this review paper is that it summarizes the current state-of-the-art IoT in industries systematically.

**L[2]: A Privacy Preserving Communication Protocol for IoT Applications in Smart**

**Homes**

**Authors:** T. Song, R. Li, B. Mei, J. Yu, X. Xing, and X. Cheng

**Description:** — The development of the Internet of Things (IoT) has made extraordinary progress in recent years in both academic and industrial fields. There are quite a few smart home systems that have been developed by major companies to achieve home automation. However, the nature of smart homes inevitably raises security and privacy concerns. In this paper, we propose an improved energy-efficient, secure, and privacy-preserving communication protocol for the smart home systems. In our proposed scheme, data transmissions within the smart home system are secured by a symmetric encryption scheme with secret keys being generated by chaotic systems. Meanwhile, we incorporate Message Authentication Codes (MAC) to our scheme to guarantee data integrity and authenticity. We also provide detailed security analysis and performance evaluation in comparison with our previous work in terms of computational complexity, memory cost, and communication overhead.

**L[3]: Developing vehicular data cloud services in the iot environment**

**Authors:**  W. He, G. Yan, and L. Da Xu

**Description:** The advances in cloud computing and internet of things (IoT) have provided a promising opportunity to resolve the challenges caused by the increasing transportation issues. We present a novel multilayered vehicular data cloud platform by using cloud computing and IoT technologies. Two innovative vehicular data cloud services, an intelligent parking cloud service and a vehicular data mining cloud service, for vehicle warranty analysis in the IoT environment are also presented. Two modified data mining models for the vehicular data mining cloud service, a Naïve Bayes model and a Logistic Regression model, are presented in detail. Challenges and directions for future work are also provided.

3

**L[4]: A privacy-preserving fog computing framework for vehicular crowdsensing networks**

**Authors:** J. Wei, X. Wang, N. Li, G. Yang, and Y. Mu

**Description:** Recently, the study of road surface condition monitoring has drawn great attention to improve traffic efficiency and road safety. As a matter of fact, this activity plays a critical role in the management of the transportation infrastructure. Trustworthiness and individual privacy affect the practical deployment of the vehicular crowdsensing network. Mobile sensing as well as contemporary applications is made use of problem solving. The fog computing paradigm is introduced to meet specific requirements, including mobility support, low latency, and location awareness. The fog-based vehicular crowdsensing network is an emerging transportation management infrastructure. Moreover, the fog computing is effective to reduce the latency and improve the quality of service. Most of the existing authentication protocols cannot help the drivers to judge a message when the authentication on the message is anonymous. In this paper, a fog-based privacy-preserving scheme is proposed to enhance the security of the vehicular crowdsensing network. Our scheme is secure with the security properties, including non-deniability, mutual authentication, integrity, forward privacy, and strong anonymity. We further analyze the designed scheme, which can not only guarantee the security requirements, but also achieve higher efficiency with regards to computation and communication compared with the existing schemes.

**L[5]: Deep learning for iot big data and streaming analytics: A survey**

**Authors:** M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani

**Description:** In the era of the Internet of Things (IoT), an enormous amount of sensing devices collect and/or generate various sensory data over time for a wide range of fields and applications. Based on the nature of the application, these devices will result in big or fast/real-time data streams. Applying analytics over such data streams to discover new information, predict future insights, and make control decisions is a crucial process that makes IoT a worthy paradigm for businesses and a quality-of-life improving technology. In this paper, we provide a thorough overview on using a class of advanced machine learning techniques, namely deep learning (DL), to facilitate the analytics and learning in the IoT domain. We start by articulating IoT data characteristics and identifying two major treatments for IoT data from a machine learning perspective, namely IoT big data analytics and IoT streaming data analytics. We also discuss why DL is a promising approach to achieve the desired analytics in these types of data and applications. The potential of using emerging DL techniques for IoT data analytics are then discussed, and its promises and challenges are introduced. We present a comprehensive background on different DL architectures and algorithms. We also analyze and summarize major

4

reported research attempts that leveraged DL in the IoT domain. The smart IoT devices that have incorporated DL in their intelligence background are also discussed. DL implementation approaches on the fog and cloud centers in support of IoT applications are also surveyed. directions for future research. At the end of each section, we highlight the lessons learned based on our experiments and review of the recent literature.

**L[6]: A novel latin-square based secret sharing for m2m communications**

**Authors:** J. Shen, T. Zhou, X. Liu, and Y.-C. Chang

**Description:** Machine-to-machine (M2M) communication, an automated communications technology for the equipment or devices, holds great promise in every corner of the modern society, such as civil transportation, smart homes, smart grids, and industrial automation. The M2M technology is still in its infancy, and further development and deployment of M2M systems hinges on establishing an efficient and secure information management system with a satisfactory security level. In this paper, we extend the idea of $(t,n)$ secret sharing for information transmission in M2M with high security and efficiency. Specifically, a secret is divided into $2k$ shares, and then, transmitted through $2k$ node-disjoint paths constructed by the Latin square. Note that in our scheme, the secret key is simultaneously transmitted along with the encrypted message through these $2k$ paths from the source node to the destination node, which greatly improves the efficiency of point-to-point communications in M2M systems. Furthermore, owing to the properties of $(t,n)$ secret sharing, the security of M2M communications is guaranteed. In addition, to avoid dishonest participants, verifiable secret sharing is supported in the proposed scheme. Sufficient theoretical proof and performance analysis demonstrate that our scheme is secure and efficient for M2M communications.

**L[7]: Machine learning in adversarial settings**

**Authors:** P. McDaniel, N. Papernot, and Z. B. Celik

**Description:** Deep learning takes advantage of large datasets and computationally efficient training algorithms to outperform other approaches at various machine learning tasks. However, imperfections in the training phase of deep neural networks make them vulnerable to adversarial samples: inputs crafted by adversaries with the intent of causing deep neural networks to misclassify. In this work, we formalize the space of adversaries against deep neural networks (DNNs) and introduce a novel class of algorithms to craft adversarial samples based on a precise understanding of the mapping between inputs and outputs of DNNs. In an application to computer vision, we show that our algorithms can reliably produce samples correctly classified by human subjects but misclassified in specific targets by a DNN with a 97% adversarial success rate while only modifying on average 4.02% of the input features per sample. We then evaluate the vulnerability of different sample classes to adversarial perturbations by defining a hardness measure. Finally, we describe preliminary work outlining defenses against adversarial samples by defining a predictive measure of distance between a benign input and a target classification.

**L[8]: Hop-by-hop message authentication and source privacy in wireless sensor networks**

**Authors:** J. Li, Y. Li, J. Ren, and J. Wu

**Description:** Message authentication is one of the most effective ways to thwart unauthorized and corrupted messages from being forwarded in wireless sensor networks (WSNs). For this reason, many message authentication schemes have been developed, based on either symmetric-key cryptosystems or public-key cryptosystems. Most of them, however, have the limitations of high computational and communication overhead in addition to lack of scalability and resilience to node compromise attacks. To address these issues, a polynomial-based scheme was recently introduced. However, this scheme and its extensions all have the weakness of a built-in threshold determined by the degree of the polynomial: when the number of messages transmitted is larger than this threshold, the adversary can fully recover the polynomial. In this paper, we propose a scalable authentication scheme based on elliptic curve cryptography (ECC). While enabling intermediate nodes authentication, our proposed scheme allows any node to transmit an unlimited number of messages without suffering the threshold problem. In addition, our scheme can also provide message source privacy. Both theoretical analysis and simulation results demonstrate that our proposed scheme is more efficient than the polynomial-based approach in terms of computational and communication overhead under comparable security levels while providing message source privacy.

**L[9]: A public key cryptosystem and a signature scheme based on discrete logarithms**

**Author:** T. ElGamal

**Description:** A new signature scheme is proposed, together with an implementation of the Diffie-Hellman key distribution scheme that achieves a public key cryptosystem. The security of both systems relies on the difficulty of computing discrete logarithms over finite fields.

**3. REQUIREMENTS SPECIFICATIONS**

**3.1 Existing system:**

In order to prevent various types of attacks in data transmission, both symmetric-key and public-key approaches have been proposed in the literature. In [12], two different message authentication protocols were proposed. The first protocol, named TESLA, is based on Message Authentication Code (MAC), and the design utilizes a one-way key chain and timed release of keys by the sender. However, the TESLA protocol requires synchronization among devices, which is difficult to implement in a large scale network. The second protocol in [12], named EMSS, is based on cryptographic hash function and public-key technique, and can achieve the security property of non-repudiation.

In [13], an interleaved hop-by-hop authentication scheme was proposed to prevent the injected false data packet attack by attackers or compromised

nodes in the network. Their scheme is symmetric-key based, and the basic idea is that multiple sensor nodes have to endorse a message (or report) using MACs in order to achieve message authentication. A similar approach was also proposed in an

independent work by Ye et al. [14]. In [15], a polynomial based approach was proposed to achieve lightweight and compromise-resilient message authentication, where messages are authenticated and verified via evaluating polynomials. In [8], Li et al. proposed a ring signature [16] based solution to achieve message authentication. Their scheme utilizes a ring signature scheme derived from the modified ElGamal signature scheme [10], and can achieve better features and performance in several aspects compared with the previous solutions. However, as we will demonstrate later, the ring signature scheme proposed in [8] has a security flaw: it allows an attacker to arbitrarily form a ring and forge a valid ring signature from an existing one. Such an attack has been considered in the literature of ring signature (e.g., [17]) and in this work we introduce a technique similar to that of [17] to fix the flaw without introducing any computation or communication overhead.

There are also a number of research works on privacy preserving user authentication (and key agreement) protocols for IoT and wireless sensor networks (WSNs) in recent years (e.g., [18], [19], [20], [21], [22], [23], [24], [25], [26]). These works focus on remote user authentication, which is related but different from the privacy preserving hop-by-hop message authentication considered in this paper. Moreover, due to the concerns on the physical security of sensor nodes and IoT devices, the research on constructing lightweight and physically secure authentication protocols for IoT and wireless sensor networks has also become a popular topic in recent years. To ensure physical layer security, Physically Unclonable Functions (PUFs) and wireless channel characteristics (such as the Link Quality Indicator (LQI)) are popular choices to enable security even if a sensor node is captured by an adversary. Several novel lightweight authentication protocols with physical security for IoT and WSNs can be found in [27][28][29].

**3.2 Disadvantages of Existing System:**

* + The system is less effective due to lack of source location privacy.
  + The system has only detection techniques and no protection techniques

**3.3 Proposed System:**

Moreover, considering the low computation power of the IoT devices, we also apply the offline/online paradigm in the design of our system. Efficiency is extremely important in practical IoT scenarios such as industrial automation, environmental

monitoring, smart grids, etc. In proposed scheme, a smart device can perform some expensive public-key operations offline (e.g., when it is idle), and only does the online computation when the message to be sent is ready. Interestingly, we find that by allowing both RSA and ElGamal type systems in our scheme, we are able to reduce the computation cost compared with the pure ElGamal scheme proposed in [8]. This may look counterintuitive since it is known that the ElGamal system (implemented using Elliptic Curve Cryptography, or ECC for short) is much faster than the RSA system. The reason of this counterintuitive fact is that in our hybrid scheme, for most of the RSA nodes, we only need to do RSA signature verification,

which is very fast since the RSA public exponent e can be very small. The proposed new SAMA scheme is compared with the previous scheme in terms of its execution time during signature generation and verification. We also implement our scheme in a laptop and in a Raspberry Pi to demonstrate its practicality.

**3.4 Advantages of Proposed System:**

* Authenticity: The receiver and each forwarder in the routing path can verify that the message is sent by a legitimate data source, which can be a specific node, or a node in a particular group.
* Integrity: The receiver and each forwarder in the routing path can verify that the message has not been altered during transmission.
* Identity and location privacy: the identity and location of the message sender is well-protected. As mentioned before, the identity and location of a node may disclose some information about the data sent by that node.

**3.5 Software and Hardware requirements:**

**H/W System Configuration:**

* Processor - Pentium –IV
* RAM - 4 GB (min)
* Hard Disk - 20 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**Software Requiremnets:**

* Operating system : Windows XP or Windows 7, Windows 8.
* Coding Language : Java – AWT, Swings, Networking
* Data Base : My Sql / MS Access.
* Documentation : MS Office
* IDE : Eclipse Galileo
* Development Kit : JDK 1.6

**4. SYSTEM STUDY**

**4.1 Feasibility Study:**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**4.2 Economical Feasibility:**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **4.3 Technical Feasibility:**

### This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**4.4 Social Feasibility:**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**5. SYSTEM DESIGN**

**5.1 System Architecture:**

1. Initialize mac for all nodes
2. Receive Data
3. Find Path using DSR Protocol
4. Find attackers
5. Send Attackers to IDS Manager
6. Find Trust and Un Trust Nodes
7. View all sensor macs

**IOT Device Source**

Init mac based on file contents

Encrypt Upload data

Init MAC confirmation

Data Received confirmation

Send Attacker Details

**Trust Manager**

1. Receive Data   
2.Find Time delay  
3.Store data Details

Find Attackers

Attack data

***Apply Entropy Anomaly Detection and prevention***

**Sinks A,B,C,D,E**

**Attacker**

**5.2 Dataflow Diagram:**

Level -0

**IOT Device Source**

Level-1

**IOT Device Source**

Router

Level-2

**IOT Device Source**

Router

No Yes

Trust Manager

Level-3

**SOURCE**

Router

No Yes

**Check Attacker Type?**

IDS Manager

**Active Attacker**

**Passive Attack**

Level-4

**SOURCE**

Router

No Yes

**Check Attacker Type?**

IDS Manager

**Active Attacker**

**Filter the False injected data**

**Passive Attack**

**Update IP Address**

Acknowledgement from Receiver

Destination

**5.3 Flow Chart:**

**Router**

**Check Node Status**

Malicious

**IOT Device Source**

Malicious

**Forward data to receiver**

No Malicious

**Find False Injected Data Attacker**

**Want to apply Filtering or En routing Tech**

No

**Send Injected Data to sink**

Yes

**Trust MANAGER**

****

True

False

**Verify MAC**

**Injected Data Found**

**Sinks**

### **NOTATION:**

### **SOURCE OR DESTINATION OF DATA**:

External sources or destinations, which may be people or organizations or other entities.

### **DATA SOURCE:**

Here the data referenced by a process is stored and retrieved.

### **PROCESS:**

People, procedures or devices that produce data. The physical component is not identified.

### **DATA FLOW:**

Data moves in a specific direction from an origin to a destination. The data flow is a “packet” of data.

## **MODELING RULES:**

There are several common modeling rules when creating DFDs:

1. All processes must have at least one data flow in and one data flow out.
2. All processes should modify the incoming data, producing new forms of outgoing data.
3. Each data store must be involved with at least one data flow.
4. Each external entity must be involved with at least one data flow.

A data flow must be attached to at least one process.

A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an [information system](http://en.wikipedia.org/wiki/Information_system). DFDs can also be used for the [visualization](http://en.wikipedia.org/wiki/Data_visualization) of [data processing](http://en.wikipedia.org/wiki/Data_processing) (structured design).On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. A DFD provides no information about the timing of processes, or about whether processes will operate in sequence or in parallel.

**5.3 UseCase Diagram**

**Upload confirmation**

**IOT Device Source**

**Router**

**Browse files**

**Assign BW**

**Receive MAC**

**Send MAC to all Nodes**

**Upload file**

**Init All Sensor Node**

**Find Gang False Injected Data**

**Apply En Routing Technique**

Sinks

**Backlog on False Injected Data Attackers**

**Receive data**

**Verify MAC in Sink**

Trust MANAGER

**Accept Data**

**Finding attackers**

**Message attacker**

**Energy attacker**

**Reject Data**

**If node malicious choose alternate path**

A **use case diagram** in the [Unified Modeling Language](http://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML) is a type of behavioral diagram defined by and created from a [Use-case analysis](http://en.wikipedia.org/wiki/Use-case_analysis). Its purpose is to present a graphical overview of the functionality provided by a system in terms of [actors](http://en.wikipedia.org/wiki/Actor_(UML)), their goals (represented as [use cases](http://en.wikipedia.org/wiki/Use_case)), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. A use case diagram is a type of behavioral diagram created from a [Use-case analysis](http://en.wikipedia.org/wiki/Use-case_analysis).

**5.4 Class Diagram:**

Members

Methods

**IOT Device Source**

Trust Manager

Router

Connect (), Forward (), (), Attacker (),attacker (), refresh (), Socket s, Server Socket

Node name, attacker name, date and time

Connect (), Receive (), Forward (), Assign\_ BW (), Attacker (),Init\_MAC

Energy, Switched nodes, Attacker status, Filtering

Methods

Browse, Upload, Reset, Init\_MAC ()

File\_ Name, Sender\_ Name, Router\_Name, Destination\_Name, MAC

Methods

Methods

Members

Members

Members

Sink Attacker

Active attacker (), get text (), nodes (), Passive Attacker (), IP ()

Data name, hash, data, IP address

Receive (),   
Verify\_MAC\_Confirm (),   
 Store (), Accept ()

Fname, store, MAC

The class diagram is the main building block of [object oriented](http://en.wikipedia.org/wiki/Object_oriented) modeling. It is used both for general [conceptual modeling](http://en.wikipedia.org/wiki/Conceptual_model) of the systematic of the application, and for detailed modeling translating the models into [programming code](http://en.wikipedia.org/wiki/Programming_code). Class diagrams can also be used for modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

In the diagram, classes are represented with boxes which contain three parts

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake

In the design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the static relations between those objects. With detailed modeling, the classes of the conceptual design are often split into a number of subclasses.

**5.5 Sequence Diagram:**

IOT Device Source

localhost

Sinks

Attacker

Trust manager

mma

Router

browser

BrowserBrowse and

Assign BW to each and every node

Init MAC To all Sensors

File Upload

Finding attacks

Message attacks

adds malicious data

File Receiving Confirmation

View MAC Details

Find False Injected Data

Message or Energy

Re Send Data

Apply En Routing Techniques

Apply Filtering Techniques

Find Gang Injecting False Data Attack

Passive attacks

Change ip address

Receive File

Receive File

Store File

Very Sink and MAC

File Sending Confirmation

Store File

Store Confirmation

**6. IMPLEMENTATION**

**6.1 Modules and their Description:**

* **IOT Device Source**

In this module, the Source browses the required file, initializes nodes with digital signature and uploads to the end user (node a, node b, node c, node d, node e, node f) via Router.

* **Router**

The Router is responsible for forwarding the data file in shortest distance to the destination; the Router consists of Group of nodes, the each and every node (n1, n2, n3,n4,n5,n6,n7,n8,n8,n10,n11,n12, n13) consist of Bandwidth and Digital Signature(MAC). If router had found any malicious or traffic node in the router then it forwards to the IDS Manager. In Router we can assign the bandwidth for the nodes and can view the node details with their tags Node Name, Sender IP, Injected data, Digital Signature, Bandwidth and status.

* **IDS Manger**

The IDS manager is nothing but Intrusion Detection System manager which is responsible to filter the malicious data and traffic data. The IDS manager decides the phases based on Router status and then decides on two phases i.e., the “Training Phase” and the “Test Phase”.

**Training Phase:**

The Normal Profile Generation module is operated in the Training Phase to generate profiles for various types of legitimate traffic records, and the generated normal profiles are stored in a database.

**Test Phase:**

The Tested Profile Generation module is used in the Test Phase to build profiles for individual observed traffic records. Then, the tested profiles are handed over to the Attack Detection module, which compares the individual tested profiles with the respective stored normal profiles.

* **Sinks**

In this module, the destination can receive the data file from the Service Provider which is sent via Router, if malicious or traffic node is found in the router then it forwards to the IDS Manager to filter the content and adds to the attacker profile.

* **Forgery Attacker and Packet Droppers**

In this module, the malicious node or the traffic node details can be identified by a threshold-based classifier is employed in the Attack Detection module to distinguish DoS attacks from legitimate traffic. The Attacker can inject the fake message and generates the signature to a particular node in the router with the help of threshold-based classifier in testing phase and then adds to the attacker profile.

**7. SOFTWARE ENVIRONMENT**

## **7.1 Java Technology:**

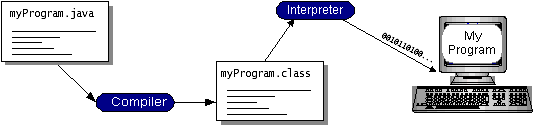
Java technology is both a programming language and a platform.

### **7.2 The Java Programming Language:**

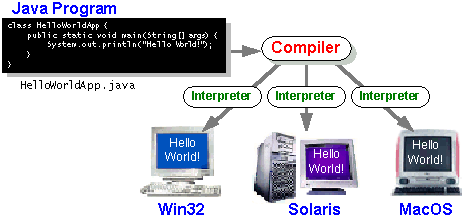
### The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

* + - Simple
    - Architecture neutral
    - Object oriented
    - Portable
    - Distributed
    - High performance
    - Interpreted
    - Multithreaded
    - Robust
    - Dynamic
    - Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



### **7.3 The Java Platform:**

A platform is the hardware or software environment in which a program runs. We’ve already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other hardware-based platforms.

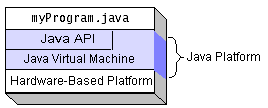
The Java platform has two components:

* The Java Virtual Machine (Java VM)
* The Java Application Programming Interface (Java API)

You’ve already been introduced to the Java VM. It’s the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that’s running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

## **What Can Java Technology Do?**

The most common types of programs written in the Java programming language are applets and applications. If you’ve surfed the Web, you’re probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

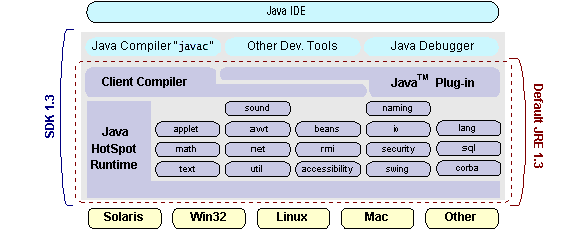
However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

* **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
* **Applets**: The set of conventions used by applets.
* **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
* **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
* **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
* **Software components**: Known as JavaBeansTM, can plug into existing component architectures.
* **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
* **Java Database Connectivity (JDBCTM)**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



### **7.4 ODBC :**

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate

list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn’t change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn’t as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

**7.5 JDBC:**

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

### **7.6 JDBC Goals:**

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. **SQL Level API**

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

1. **SQL Conformance**

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

1. **JDBC must be implemental on top of common database interfaces**The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.
2. **Provide a Java interface that is consistent with the rest of the Java system**

Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

1. **Keep it simple**

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

1. **Use strong, static typing wherever possible**

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

1. **Keep the common cases simple**

Because more often than not, the usual SQL calls used by the programmer are simple SELECT’s, INSERT’s, DELETE’s and UPDATE’s, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

Finally we decided to proceed the implementation using Java Networking.

And for dynamically updating the cache table we go for MS Access database.

Java ha two things: a programming language and a platform.

Java is a high-level programming language that is all of the following

Simple Architecture-neutral

Object-oriented Portable

Distributed High-performance

Interpreted multithreaded

Robust Dynamic

Secure

Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.

**Java Program**

**Compilers**

**Interpreter**

**My Program**

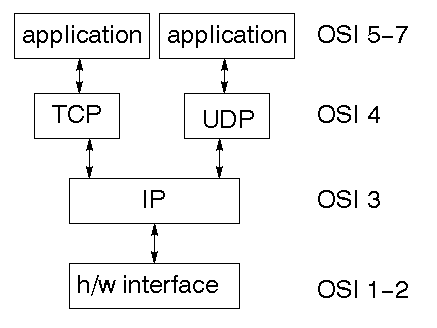
You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

## **7.6 Networking**

### TCP/IP stack

The TCP/IP stack is shorter than the OSI one:



TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

### IP datagram’s

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

### **UDP**

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

### **TCP**

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

### Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

### **Network address**

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

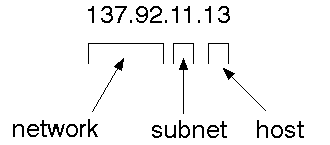
### **Subnet address**

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

**Host address**

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

### **Total address**



The 32 bit address is usually written as 4 integers separated by dots.

### **Port addresses**

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

### **Sockets**

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call socket. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

#include <sys/types.h>

#include <sys/socket.h>

int socket(int family, int type, int protocol);

Here "family" will be AF\_INET for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

**7.7 JFree Chart:**

JFree Chart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFree Chart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

JFreeChart is "open source" or, more specifically, [free software](http://www.gnu.org/philosophy/free-sw.html). It is distributed under the terms of the [GNU Lesser General Public Licence](http://www.gnu.org/licenses/lgpl.html) (LGPL), which permits use in proprietary applications.

## **1. Map Visualizations**

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart;

Testing, documenting, testing some more, documenting some more.

## **2. Time Series Chart Interactivity**

Implement a new (to JFreeChart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

## **3. Dashboards**

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

## **4. Property Editors**

The property editor mechanism in JFreeChart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

**7.8 J2ME (Java 2 Micro edition):**

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems." Announced in June 1999 at the JavaOne Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

**1. General J2ME architecture**



J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the The profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. We'll cover profiles in depth in the The following graphic depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

**2.Developing J2ME applications**

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role pre verification plays in this process.

**3.Design considerations for small devices**

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process. Here are some design strategies to consider:

\* Keep it simple. Remove unnecessary features, possibly making those features a separate, secondary application.

\* Smaller is better. This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.

\* Minimize run-time memory use. To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector. You should manage the memory efficiently yourself by setting object references to null when you are finished with them. Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an as-needed basis. Other ways of reducing overall and peak memory use on small devices are to release resources quickly, reuse objects, and avoid exceptions.

**4.Configurations overview**

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

\* **Connected Limited Device Configuration (CLDC)** is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC. CLDC is also the configuration that we will use for developing our drawing tool application. An example of a small wireless device running small applications is a Palm hand-held computer.

\* **Connected Device Configuration (CDC)** is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

**5.J2ME profiles**

**What is a J2ME profile?**

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both KJava and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

A skeleton profile upon which you can create your own profile, the Foundation Profile, is available for CDC.

**Profile 1: KJava**

KJava is Sun's proprietary profile and contains the KJava API. The KJava profile is built on top of the CLDC configuration. The KJava virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine. KJava contains a Sun-specific API that runs on the Palm OS. The KJava API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT). However, because it is not a standard J2ME package, its main package is com.sun.kjava. We'll learn more about the KJava API later in this tutorial when we develop some sample applications.

**Profile 2: MIDP**

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application

development. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

\* java.lang

\* java.io

\* java.util

\* javax.microedition.io

\* javax.microedition.lcdui

\* javax.microedition.midlet

\* javax.microedition.rms

**8.SYSTEM TESTING**

### **8.1 Testing Methodologies:**

### The following are the Testing Methodologies:

* Unit Testing.
* Integration Testing.
* User Acceptance Testing.
* Output Testing.
* Validation Testing.

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module’s control structure to

ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

**Integration Testing**

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

**User Acceptance Testing**

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

**Output Testing**

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

**Validation Checking**

Validation checks are performed on the following fields.

**Text Field:**

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

**Numeric Field:**

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

**Preparation of Test Data**

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

**Using Live Test Data:**

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

**Using Artificial Test Data:**

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

**8.2 User Training:**

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

**8.3** **Maintenance:**

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user’s requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

**8.4 Testing Strategies :**

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation.A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

**SYSTEM TESTING:**

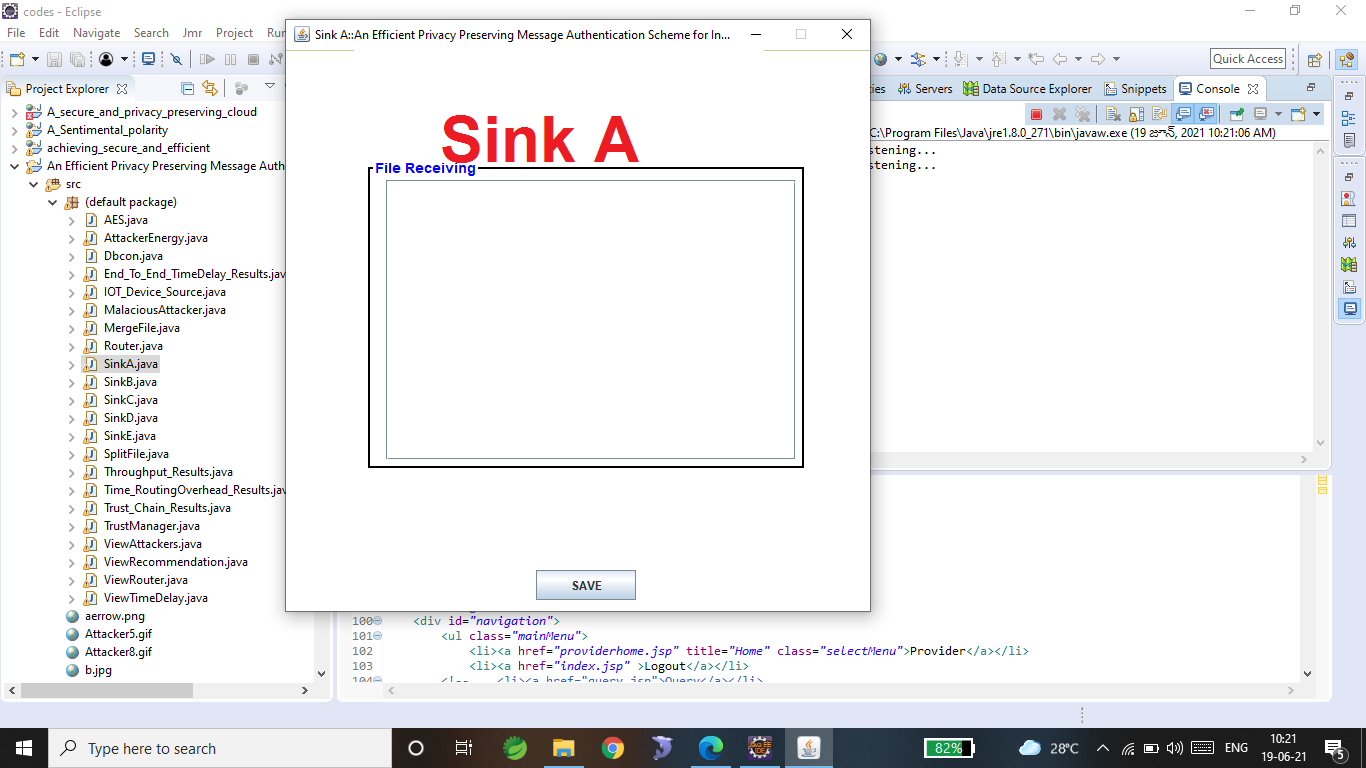
Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

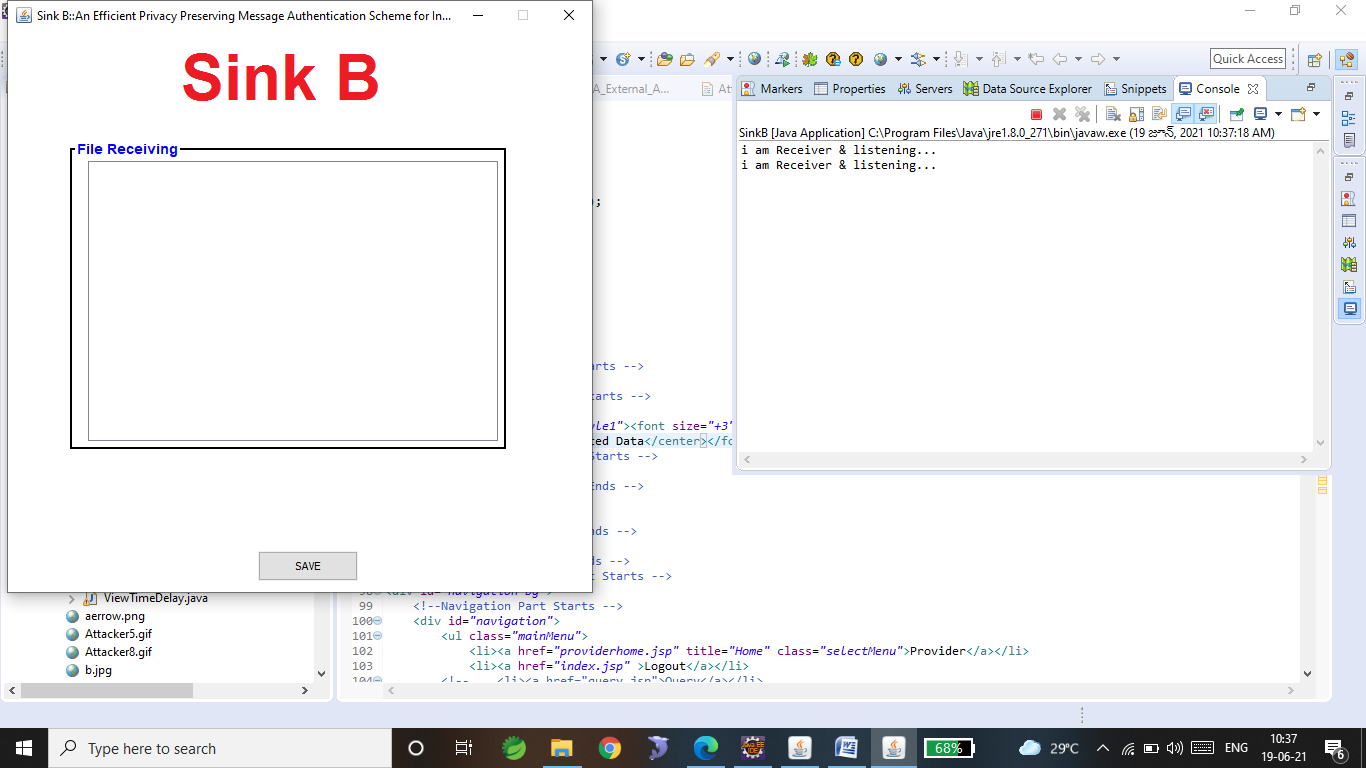
**Unit Testing:**

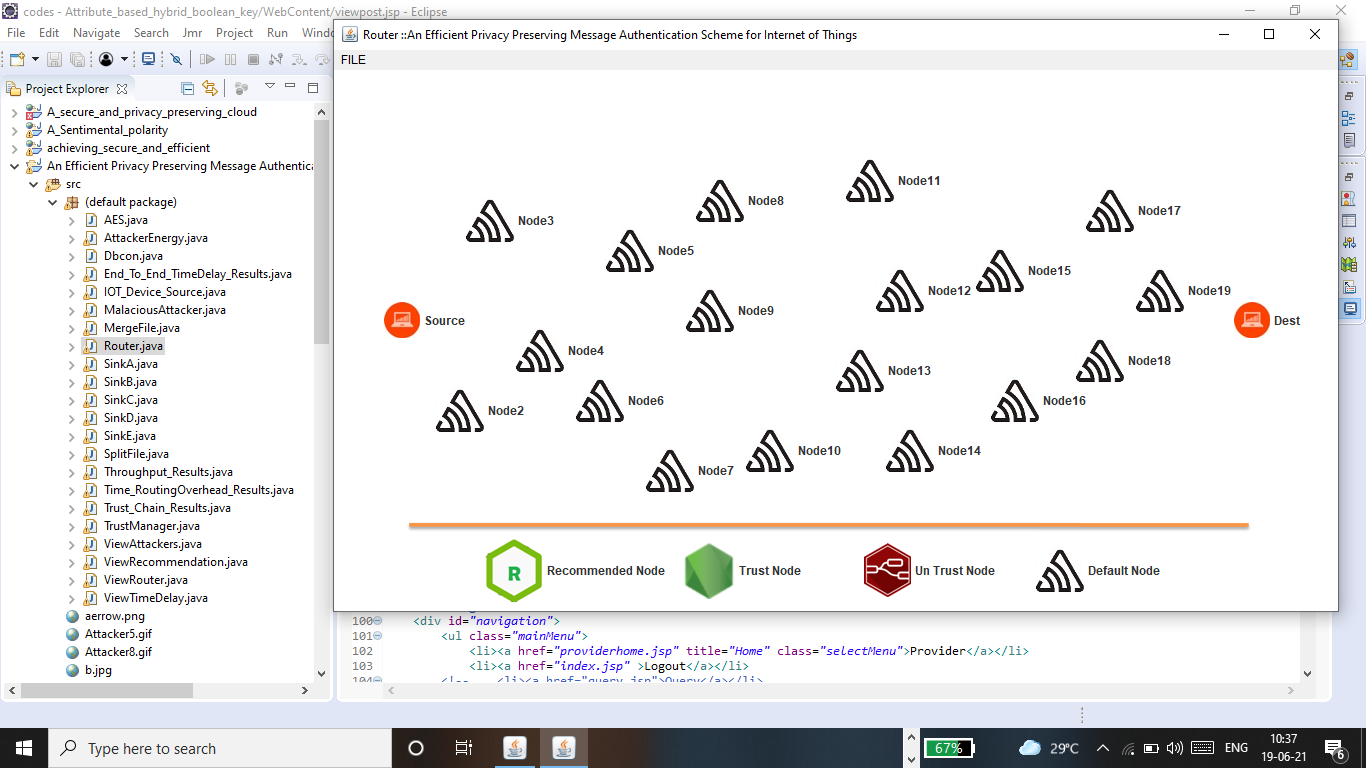
In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

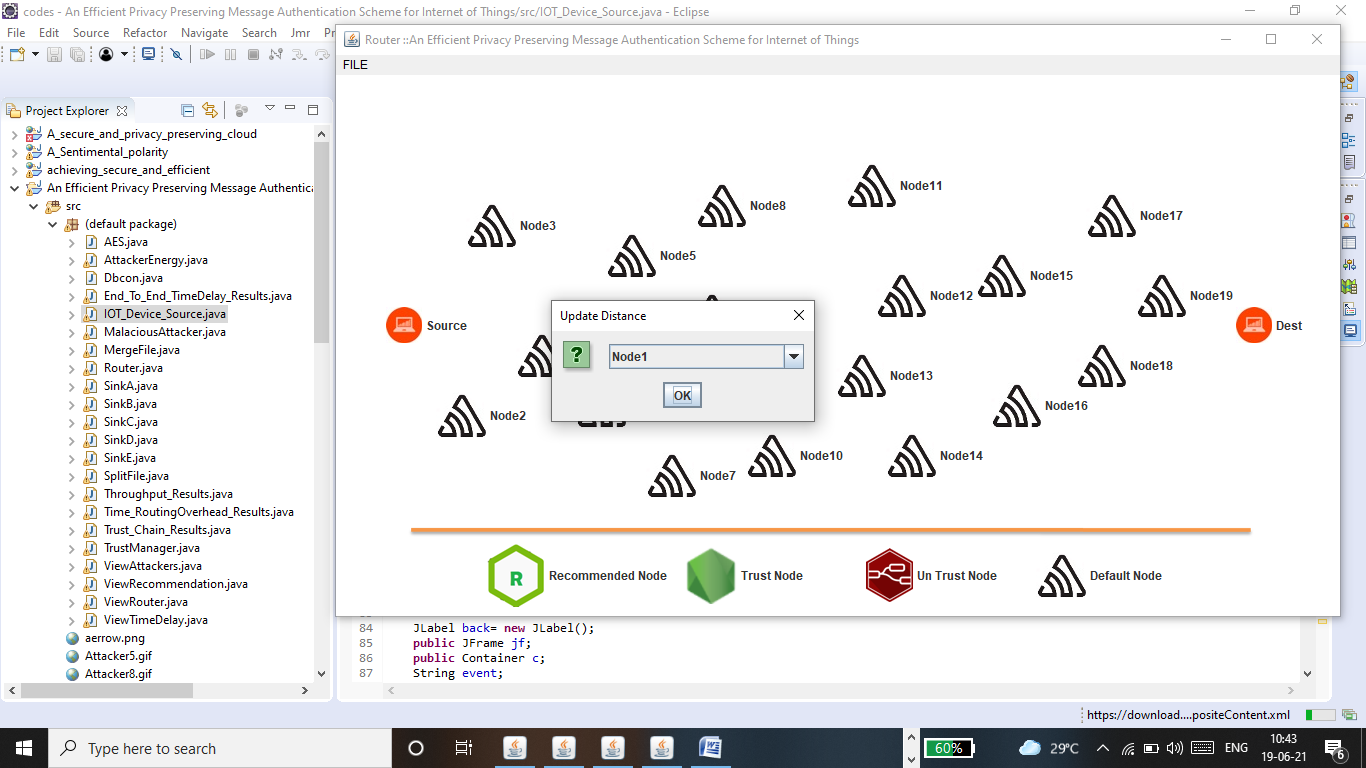
In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this.The future holds a lot to offer to the development and refinement of this project.

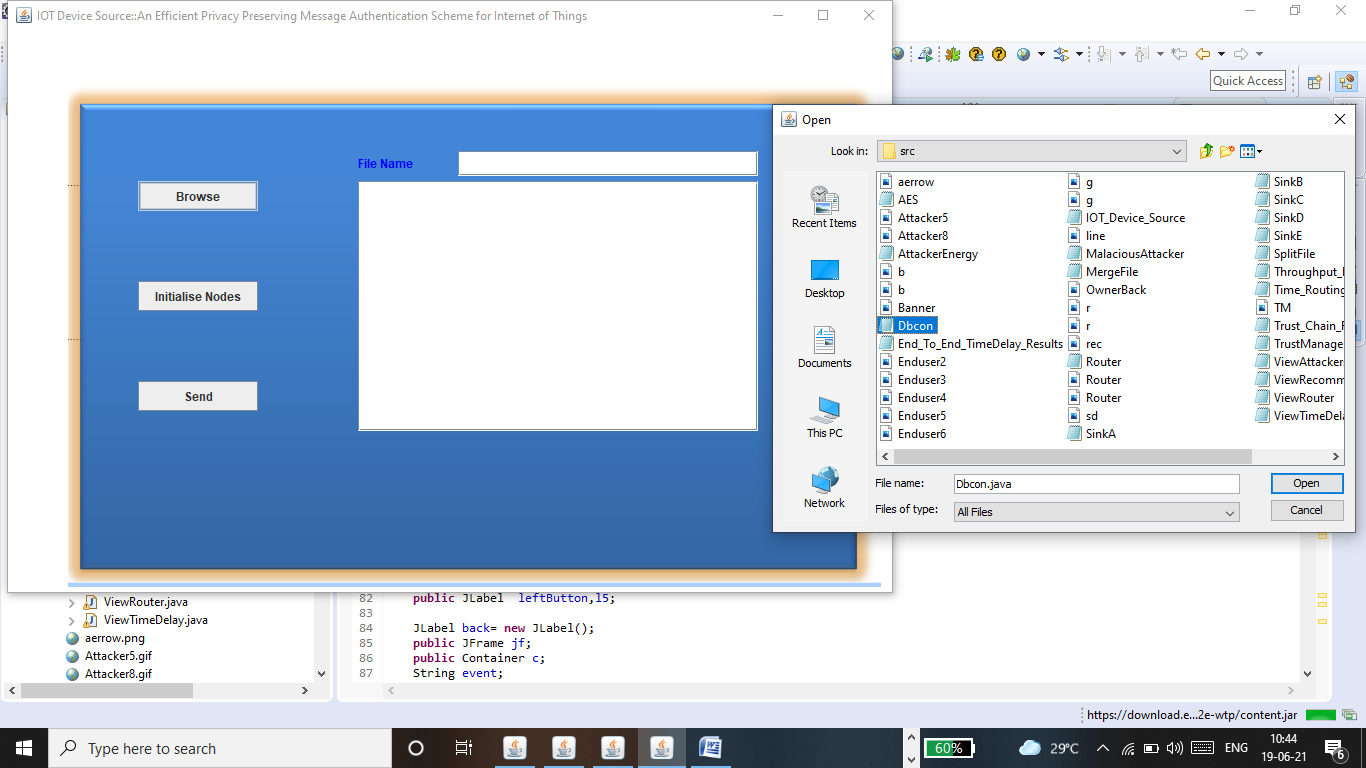
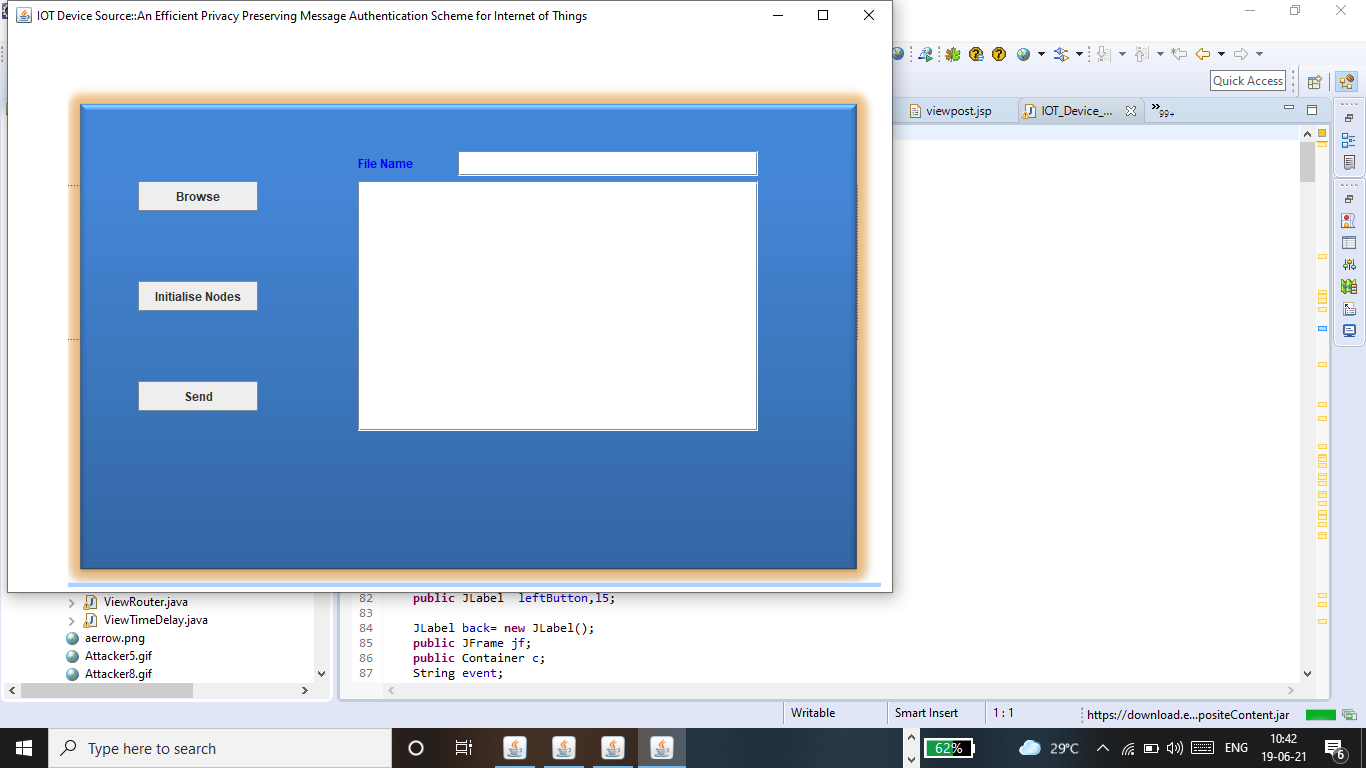
1. **OUTPUT SCREENS**

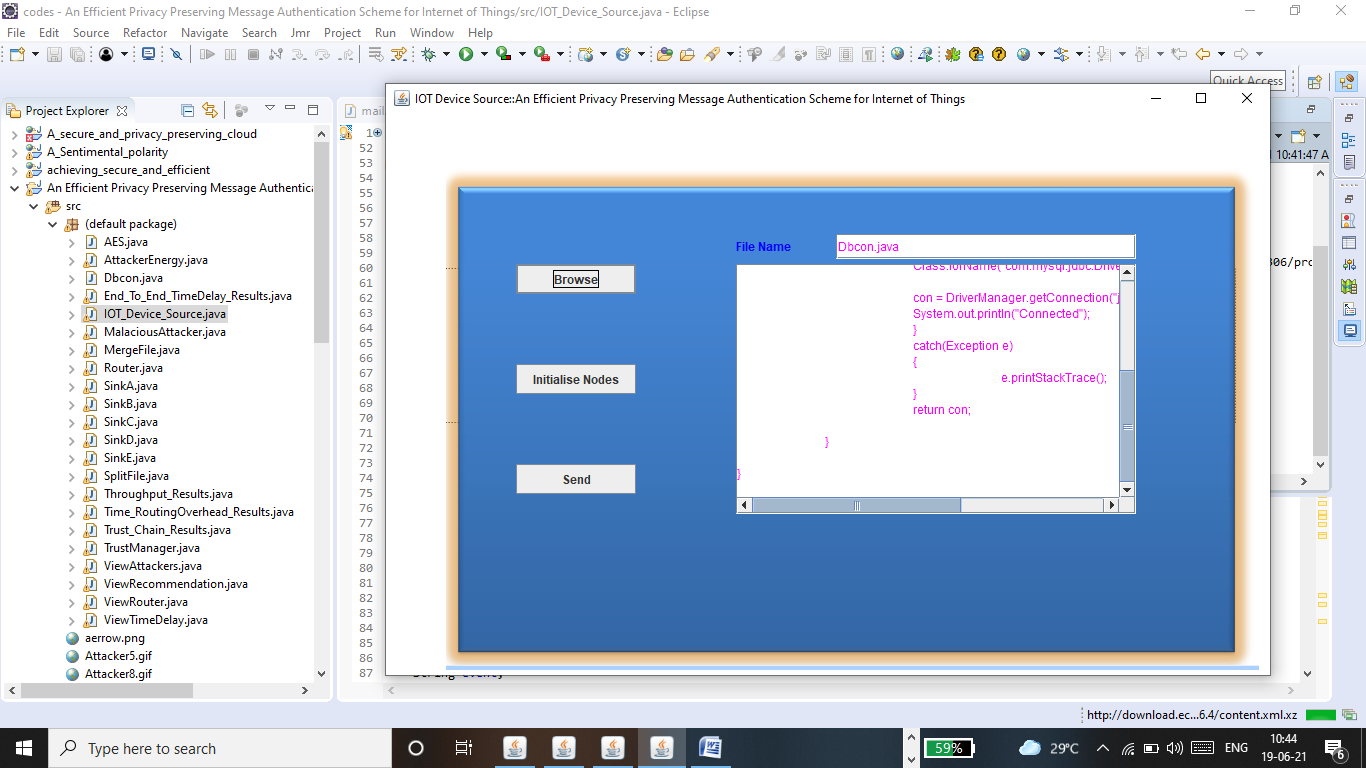


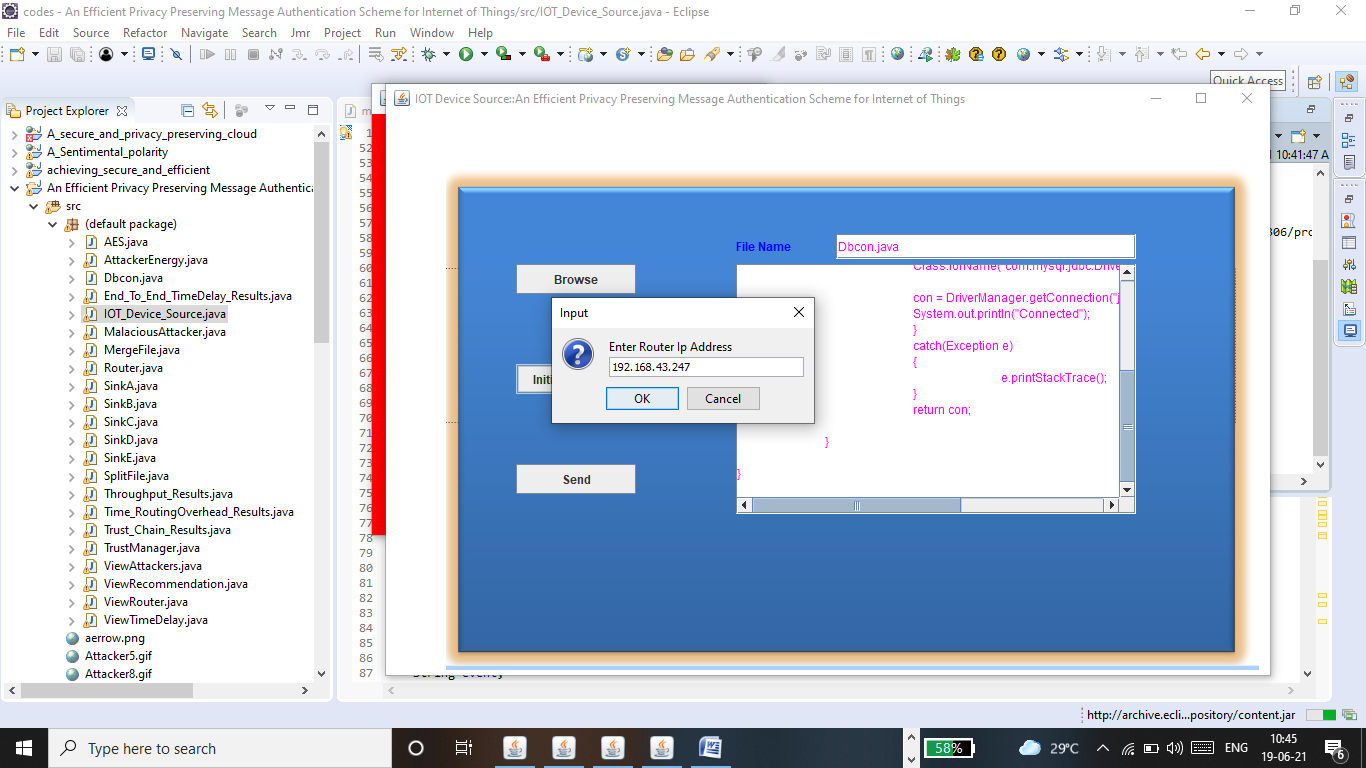


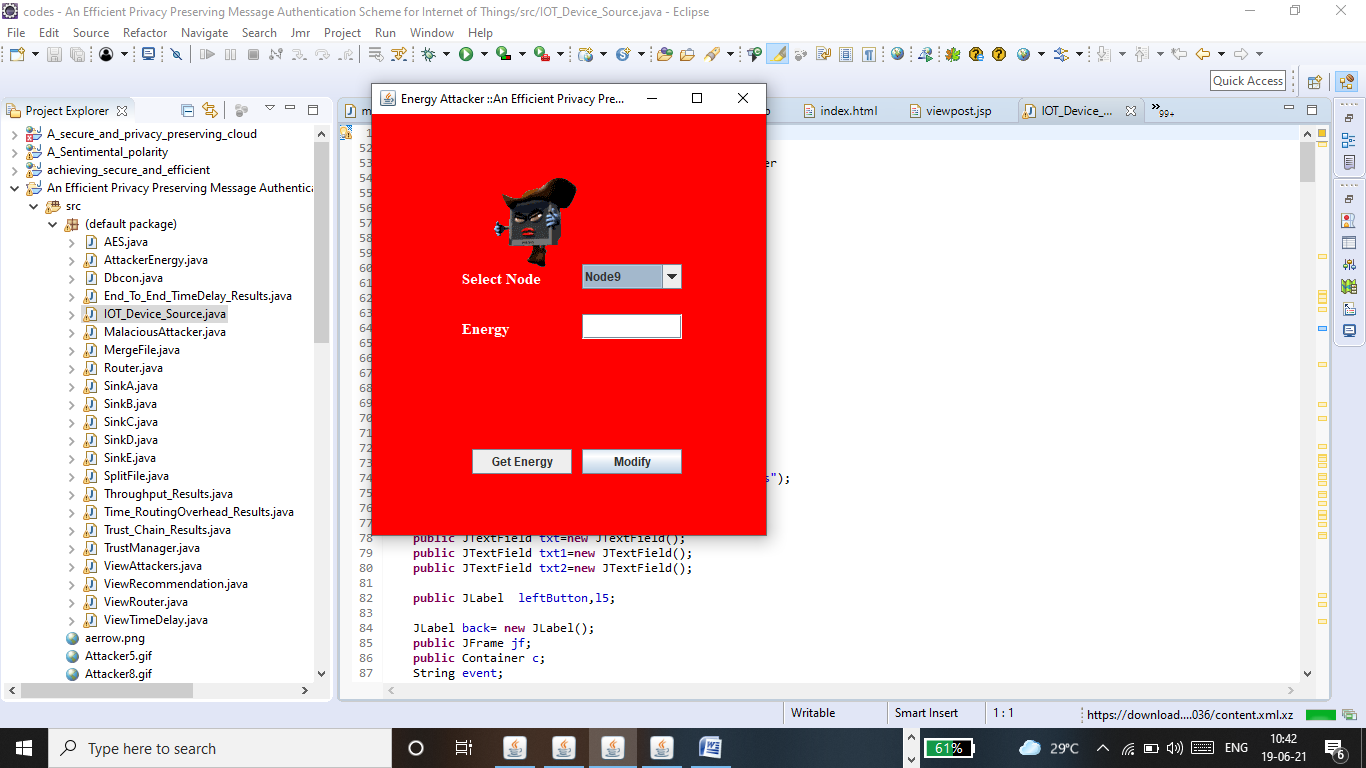


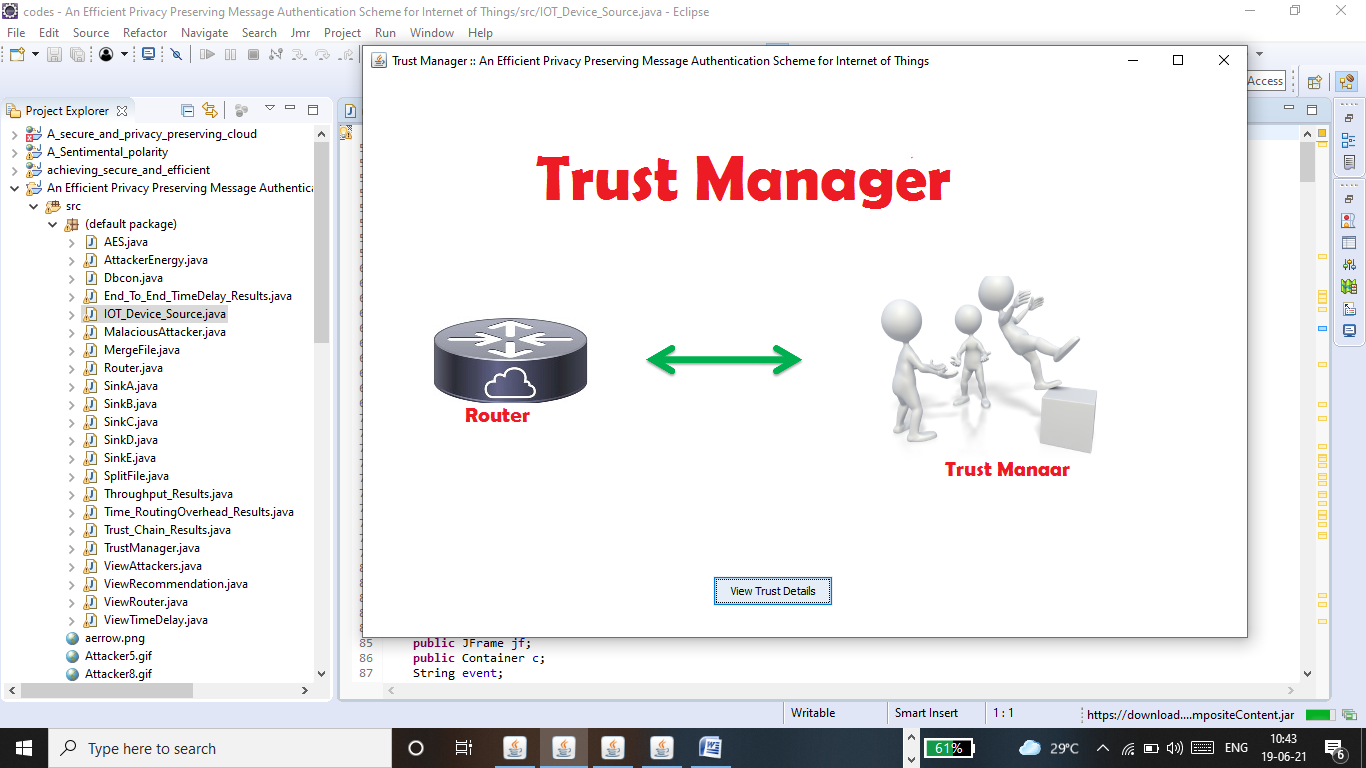












**10. CONCLUSION**

In this paper, we revisited a privacy-preserving message authentication scheme and showed a security weakness in the scheme. We also provided a solution to fix the problem without introducing any overhead. In order to provide better practicality in IoT consisting of different types of smart devices, we also proposed a new privacy-preserving message authentication scheme that allows IoT devices to use different security systems and parameters. Moreover, we applied the offline/online computation technique to improve the efficiency and scalability of the proposed scheme, which makes it more practical compared with the previous solution.

**11.REFERENCE**

[1] L. Da Xu, W. He, and S. Li, “Internet of things in industries: A survey, ”IEEE Transactions on industrial informatics, vol. 10, no. 4, pp. 2233–2243, 2014.

[2] T. Song, R. Li, B. Mei, J. Yu, X. Xing, and X. Cheng, “A privacy preserving communication protocol for IOT applications in smart homes,”IEEE Internet of Things Journal, vol. 4, no. 6, pp. 1844–1852, 2017.

[3] W. He, G. Yan, and L. Da Xu, “Developing vehicular data cloud services in the iot environment,” IEEE Transactions on Industrial Informatics, vol. 10, no. 2, pp. 1587–1595, 2014.

[4] J. Wei, X. Wang, N. Li, G. Yang, and Y. Mu, “A privacy-preserving fog computing framework for vehicular crowdsensing networks,” IEEE Access, vol. 6, pp. 43 776–43 784, 2018.

[5] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, “Deep learning for iot big data and streaming analytics: A survey,” IEEE Communications Surveys Tutorials, vol. 20, no. 4, pp. 2923–2960, 2018.

[6] J. Shen, T. Zhou, X. Liu, and Y.-C. Chang, “A novel latin-squarebased secret sharing for m2m communications,” IEEE Transactions on Industrial Informatics, vol. 14, no. 8, pp. 3659–3668, 2018.

[7] P. McDaniel, N. Papernot, and Z. B. Celik, “Machine learning in adversarial settings,” IEEE Security Privacy, vol. 14, no. 3, pp. 68–72, 2016.

[8] J. Li, Y. Li, J. Ren, and J. Wu, “Hop-by-hop message authentication and source privacy in wireless sensor networks,” Parallel and Distributed Systems, IEEE Transactions on, vol. 25, no. 5, pp. 1223–1232, 2014.

[9] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” in Advances in Cryptology - CRYPTO ’84, 1985, pp. 10–18.

[10] D. Pointcheval and J. Stern, “Security proofs for signature schemes,” in Advances in Cryptology - EUROCRYPT ’96, 1996, pp. 387–398.

[11] R. L. Rivest, A. Shamir, and L. M. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Commun. ACM, vol. 21, no. 2, pp. 120–126, 1978.

[12] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, “Efficient authentication and signing of multicast streams over lossy channels,” in Security and Privacy (S&P), IEEE Symposium on, 2000, pp. 56–73.

[13] S. Zhu, S. Setia, S. Jajodia, and P. Ning, “An interleaved hop-by hop authentication scheme for filtering of injected false data in sensor networks,” in Security and Privacy (S&P), IEEE Symposium on, 2004, pp. 259–271.

[14] F. Ye, H. Luo, S. Lu, and L. Zhang, “Statistical en-route filtering of injected false data in sensor networks,” Selected Areas in Communications, IEEE Journal on, vol. 23, no. 4, pp. 839–850, 2005.

[15] W. Zhang, N. Subramanian, and G. Wang, “Lightweight and compromise-resilient message authentication in sensor networks,” in The 27th IEEE Conference on Computer Communications (INFOCOM),2008.

[16] R. L. Rivest, A. Shamir, and Y. Tauman, “How to leak a secret,” in Advances in Cryptology - ASIACRYPT 2001, 2001, pp. 552–565.

[17] E. Fujisaki and K. Suzuki, “Traceable ring signature,” in InternationalWorkshop on Public Key Cryptography. Springer, 2007, pp. 181–200.

[18] D. He, N. Kumar, and N. Chilamkurti, “A secure temporal-credentialbased mutual authentication and key agreement scheme with pseudo identity for wireless sensor networks,” Information Sciences, vol. 321,pp. 263–277, 2015.

[19] Q. Jiang, J. Ma, F. Wei, Y. Tian, J. Shen, and Y. Yang, “An untraceable temporal-credential-based two-factor authentication scheme using ecc for wireless sensor networks,” Journal of Network and Computer Applications, vol. 76, pp. 37–48, 2016.

[20] S. Roy, S. Chatterjee, A. K. Das, S. Chattopadhyay, S. Kumari, and M. Jo, “Chaotic map-based anonymous user authentication scheme with user biometrics and fuzzy extractor for crowdsourcing internet of things,”IEEE Internet of Things Journal, vol. 5, no. 4, pp. 2884–2895, 2017.

[21] X. Li, J. Niu, M. Z. A. Bhuiyan, F. Wu, M. Karuppiah, and S. Kumari, “A robust ecc-based provable secure authentication protocol with privacy preserving for industrial internet of things,” IEEE Transactions on Industrial Informatics, vol. 14, no. 8, pp. 3599–3609, 2017.

[22] M. N. Aman, K. C. Chua, and B. Sikdar, “Secure data provenance for the internet of things,” in Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security. ACM, 2017, pp. 11–14.

[23] X. Li, J. Niu, S. Kumari, F. Wu, A. K. Sangaiah, and K.-K. R. Choo, “A three-factor anonymous authentication scheme for wireless sensor networks in internet of things environments,” Journal of Network and Computer Applications, vol. 103, pp. 194–204, 2018.

[24] D. Wang, W. Li, and P. Wang, “Measuring two-factor authentication schemes for real-time data access in industrial wireless sensor networks,”IEEE Transactions on Industrial Informatics, vol. 14, no. 9, pp. 4081– 4092, 2018.

[25] Z. Cai, Z. He, X. Guan, and Y. Li, “Collective data-sanitization for preventing sensitive information inference attacks in social networks,” IEEE Transactions on Dependable and Secure Computing, vol. 15, no. 4,pp. 577–590, 2018.