**CHAPTER 1**

**INTRODUCTION**

Deferral or delay tolerant systems (DTNs) are regularly seen in rising applications, for example, crisis reaction, exceptional activities, brilliant situations, natural surroundings observing, and vehicular specially appointed systems where different hubs take an interest in assemble interchanges to accomplish a typical mission. The center normal for DTNs is that there is no certification of end-to-end network, along these lines causing high postponement or interruption because of natural attributes (e.g., remote medium, asset requirements, or high versatility) or deliberately getting into mischief hubs (e.g., malignant or selﬁsh). Overseeing trust efﬁciently and viably is basic to encouraging participation or coordinated effort and basic leadership errands in DTNs while meeting framework objectives, for example, dependability, accessibility, nature of administration (QoS), as well as versatility. Precise trust assessment is particularly testing in DTN situations since hubs are scantily scattered and don't frequently experience each other. Consequently, experience based proof trade among hubs may not be constantly conceivable. The absence of direct collaboration involvement in DTN situations upsets nonstop confirmation gathering and can bring about inaccurate put stock in estimation, prompting poor application execution.

In this project we will simulate the Provest Model by considering nodes, which are devices that are inside a Delay Tolerant Network. This project will also considers two intermediate nodes that are used to calculate the trust in the Provenance model. A source and a destination communicates by source sending an encrypted file and the destination receives and decrypts it. After decrypting, destination calculates the trust of the source by considering intermediates that are inside the network.

**1.1 COMPUTER SECURITY**

PC security, otherwise called cybersecurity or IT security, is the insurance of data frameworks from robbery or harm to the equipment, the product, and to the data on them, and from interruption or confusion of the administrations they give.

It incorporates controlling physical access to the equipment, and additionally ensuring against hurt that may come through system access, information and code infusion, and because of misbehavior by administrators, regardless of whether purposeful, unplanned, or because of them being deceived into going astray from secure methodology.

The field is of developing significance because of the expanding dependence on PC frameworks in many social orders and the development of "shrewd" gadgets, including cell phones, TVs and small gadgets as a component of the Internet of Things – and of the Internet and remote system, for example, Bluetooth and Wi-Fi. A few associations are swinging to huge information stages, for example, Apache Hadoop, to broaden information openness and machine figuring out how to distinguish progressed steady dangers.

A weakness is a framework helplessness or imperfection, and numerous vulnerabilities are recorded in the Common Vulnerabilities and Exposures (CVE) database and defenselessness administration is the repeating routine with regards to distinguishing, grouping, remediating, and relieving vulnerabilities as they are found. An exploitable defenselessness is one for which no less than one working assault or "adventure" exists. To secure a PC framework, it is imperative to comprehend the assaults that can be made against it, and these dangers can normally be grouped a few kinds.

**1.2 SECURITY THREATS**

# **1.2.1 Backdoors**

### A secondary passage in a PC framework, a cryptosystem or a calculation, is any mystery technique for bypassing typical verification or security controls. They may exist for a few reasons, including by unique outline or from poor arrangement. They may likewise have been included later by an approved gathering to permit some real access, or by an assailant for vindictive reasons; however paying little mind to the intentions in their reality, they make a helplessness.

**1.2.2 Denial-of-benefit assault**

Foreswearing of administration assaults are intended to make a machine or system asset inaccessible to its proposed clients. Assailants can refuse assistance to singular casualties, for example, by purposely entering a wrong secret word enough back to back circumstances to cause the casualty record to be bolted, or they may over-burden the abilities of a machine or system and square all clients immediately. While a system assault from a solitary IP address can be obstructed by including another firewall control, numerous types of Distributed foreswearing of administration (DDoS) assaults are conceivable, where the assault originates from a substantial number of focuses – and safeguarding is significantly more troublesome. Such assaults can begin from the zombie PCs of a botnet, however a scope of different procedures are conceivable including reflection and intensification assaults, where honest frameworks are tricked into sending movement to the casualty.

**1.2.3 Direct-get to assaults**

An unapproved client increasing physical access to a PC is in all probability ready to specifically duplicate information from it. They may likewise bargain security by making working framework changes, introducing programming worms, keyloggers, secret listening gadgets or utilizing remote mice. Notwithstanding when the framework is ensured by standard safety efforts, these might have the capacity to be by passed by booting another working framework or apparatus from a CD-ROM or other bootable media. Circle encryption and Trusted Platform Module are intended to keep these assaults.

**1.2.4 Eavesdropping**

Spying is the demonstration of secretly tuning in to a private discussion, commonly between has on a system. For example, projects, for example, Carnivore and Narus Insight have been utilized by the FBI and NSA to listen stealthily on the frameworks of network access suppliers. Indeed, even machines that work as a shut framework (i.e., with no contact to the outside world) can be listened in upon by means of observing the swoon electro-attractive transmissions created by the equipment; TEMPEST is a particular by the NSA alluding to these assaults.

**1.2.5 Spoofing**

Satirizing, when all is said in done, is a fake or vindictive practice in which correspondence is sent from an obscure source masked as a source known to the beneficiary. Parodying is most pervasive in correspondence instruments that do not have an abnormal state of security.

**1.2.6 Tampering**

Altering depicts a malevolent adjustment of items. Purported "Abhorrent Maid" assaults and security administrations planting of reconnaissance ability into switches are illustrations.

**1.2.7 Privilege heightening**

Benefit heightening portrays a circumstance where an assailant with some level of limited access can, without approval, lift their benefits or access level. Along these lines, for instance a standard PC client might have the capacity to trick the framework into giving them access to confined information; or even to "wind up root" and have full unhindered access to a framework.

**1.2.8 Phishing**

Phishing is the endeavour to get touchy data, for example, usernames, passwords, and cr card subtle elements straightforwardly from clients. Phishing is commonly completed by email ridiculing or texting, and it regularly guides clients to enter points of interest at a phony site whose look and feel are relatively indistinguishable to the honest to goodness one. Going after a casualty's trusting, phishing can be delegated a type of social designing.

**1.2.9 Clickjacking**

Clickjacking, otherwise called "UI change assault or User Interface review assault", is a vindictive strategy in which an aggressor traps a client into tapping on a catch or connection on another site page while the client planned to tap on the best level page. This is finished utilizing numerous straightforward or hazy layers. The assailant is essentially "commandeering" the snaps implied for the best level page and steering them to some other unessential page, undoubtedly possessed by another person. A comparable system can be utilized to commandeer keystrokes. Precisely drafting a blend of templates, iframes, catches and message boxes, a client can be driven into trusting that they are composing the secret key or other data on some bona fide page while it is being directed into an imperceptible edge controlled by the assailant.

A famous and gainful digital trick includes counterfeit CEO messages sent to bookkeeping and fund divisions. In mid 2016, the FBI detailed that the trick has fetched US organizations more than $2bn in around two years. In May 2016, the Milwaukee Bucks NBA group was the casualty of this sort of digital trick with a culprit imitating the group's leader Peter Feigin, bringing about the handover of all the group's workers' 2015 W-2 tax documents.

**1.3 SYSTEMS AT RISK**

Frameworks in danger Computer security is basic in any industry which utilizes PCs. At present, most electronic gadgets, for example, PCs, PCs and phones accompany worked in firewall security programming, yet in spite of this, PCs are not 100 percent exact and tried and true to ensure our information (Smith, Grabosky and Urbas, 2004). There are numerous methods for hacking into PCs. It should be possible through a system framework, clicking into obscure connections, associating with new Wi-Fi, downloading programming and records from dangerous locales, control utilization, electromagnetic radiation waves, and some more. Be that as it may, PCs can be ensured through well-constructed programming and equipment. By having solid inner associations of properties, programming many-sided quality can avert programming accident and security disappointment.

**1.3.1 Financial Systems**

Sites and applications that acknowledge or store charge card numbers, investment funds, and ledger data are unmistakable hacking targets, on account of the potential for prompt monetary benefit from exchanging cash, making buys, or offering the data on the underground market. In-store installment frameworks and ATMs have additionally been messed with to assemble client account information and PINs.

**1.3.2 Utilities and industrial equipment**

PCs control capacities at numerous utilities, including coordination of broadcast communications, the power framework, atomic power plants, and valve opening and shutting in water and gas systems. The Internet is a potential assault vector for such machines if associated, yet the Stuxnet worm showed that even hardware controlled by PCs yet not associated with the Internet can be defenseless against physical harm caused by malevolent summons sent to mechanical gear (all things considered uranium advancement axes) which are contaminated by means of removable media. In 2014, the Computer Emergency Readiness Team, a division of the Department of Homeland Security, researched 79 hacking occurrences at vitality organizations. Vulnerabilities in brilliant meters (a considerable lot of which utilize nearby radio or cell interchanges) can cause issues with charging extortion.

**1.3.3 Aviation**

The avionics business is extremely dependent on a progression of complex framework which could be assaulted. A straightforward power blackout at one airplane terminal can cause repercussions around the world, a great part of the framework depends on radio transmissions which could be upset, and controlling air ship over seas is particularly unsafe in light of the fact that radar observation just stretches out 175 to 225 miles seaward. There is likewise potential for assault from inside a flying machine. The results of an effective assault extend from loss of classification to loss of framework honesty, which may prompt more genuine concerns, for example, exfiltration of information, system and aviation authority blackouts, which thus can prompt air terminal terminations, loss of air ship, loss of traveler life, harms on the ground and to transportation foundation. A fruitful assault on a military aeronautics framework that controls weapons could have considerably more genuine results.

**1.3.4 Consumer Devices**

Work stations and workstations are normally tainted with malware either to accumulate passwords or money related record data, or to build a botnet to assault another objective. PDAs, tablet PCs, savvy watches, and other cell phones, for example, Quantified Self gadgets like action trackers have additionally moved toward becoming targets and a significant number of these have sensors, for example, cameras, mouthpieces, GPS beneficiaries, compasses, and accelerometers which could be abused, and may gather individual data, including delicate wellbeing data. Wi-fi, Bluetooth, and mobile phone arranges on any of these gadgets could be utilized as assault vectors, and sensors may be remotely actuated after a fruitful rupture. Home mechanization gadgets, for example, the Nest indoor regulator are likewise potential targets.

**1.3.5 Large Corporations**

Substantial companies are normal targets. Much of the time this is gone for monetary profit through fraud and includes information breaks, for example, the loss of a large number of customers' cr card subtle elements by Home Depot, Staples, and Target Corporation. Restorative records have been focused for use as a rule distinguish robbery, medical coverage misrepresentation, and mimicking patients to acquire doctor prescribed medications for recreational purposes or resale. Not all assaults are monetarily persuaded notwithstanding; for instance, security firm HB Gary Federal endured a genuine arrangement of assaults in 2011 from hacktivist assemble Anonymous in countering for the association's CEO guaranteeing to have penetrated their gathering, and Sony Pictures was assaulted in 2014 where the intention seems to have been to humiliate with information breaks, and handicapped person the organization by wiping workstations and servers.

**1.3.6 Automobiles**

Vehicles If get to is picked up to an auto's interior controller region organize, it is conceivable to incapacitate the brakes and turn the guiding wheel. Automated motor planning, journey control, electronically monitored slowing mechanisms, safety belt tensioners, entryway locks, airbags and propelled driver help frameworks make these disturbances conceivable, and self-driving autos go much further. Associated autos may utilize wi-fi and bluetooth to speak with locally available shopper gadgets, and the wireless system to contact attendant and crisis help benefits or get navigational or stimulation data; every one of these systems is a potential section point for malware or an assailant. Specialists in 2011 were even ready to utilize a noxious conservative plate in an auto's stereo framework as an effective assault vector, and autos with worked in voice acknowledgment or remote help highlights have installed amplifiers which could be utilized for listening in. A 2015 report by U.S. Congressperson Edward Markey scrutinized producers' safety efforts as deficient, and featured protection worries about driving, area, and symptomatic information being gathered, which is defenseless against manhandle by the two makers and programmers.

**1.3.7 Government**

Government and military PC frameworks are normally assaulted by activists and outside forces. Neighborhood and local government foundation, for example, movement light controls, police and insight office correspondences, faculty records, understudy records and budgetary frameworks are likewise potential focuses as they are presently all to a great extent modernized. Identifications and government ID cards that control access to offices which utilize RFID can be defenseless against cloning.

**1.3.8 Internet of Things and other Physical vulnerabilities**

The Internet of Things (IoT) is the system of physical protests, for example, gadgets, vehicles, structures and that are implanted with hardware, programming, sensors, and system network that empowers them to gather and trade information - and concerns have been raised this is being produced without fitting thought of the security challenges included. While the IoT makes open doors for more straightforward coordination of the physical world into PC based frameworks, it additionally gives chances to abuse. As the Internet of Things spreads broadly, digital assaults are probably going to end up an inexorably physical (as opposed to just virtual) risk. On the off chance that a front entryway's bolt is associated with the Internet, and can be bolted/opened from a telephone, at that point a criminal could enter the home at the press of a catch from a stolen or hacked telephone. Individuals could remain to lose substantially more than their cr card numbers in a world controlled by IoT-empowered gadgets. Cheats have likewise utilized electronic intends to dodge non-Internet-associated lodging entryway locks. Restorative gadgets have either been effectively assaulted or had possibly lethal vulnerabilities illustrated, incorporating both in-healing facility demonstrative gear and embedded gadgets including pacemakers and insulin pumps.

**1.4 Impact of Security Breaches**

Genuine budgetary harm has been caused by security ruptures, but since there is no standard model for evaluating the cost of an occurrence, the main information accessible is what is made open by the associations included. A few PC security counseling firms deliver evaluations of aggregate overall misfortunes owing to infection and worm assaults and to threatening advanced acts all in all. The 2003 misfortune assesses by these organizations run from $13 billion (worms and infections just) to $226 billion (for all types of undercover assaults). The unwavering quality of these assessments is frequently tested; the fundamental procedure is essentially episodic. In any case, sensible assessments of the budgetary cost of security breaks can enable associations to settle on sound venture choices. As per the exemplary Gordon-Loeb Model investigating the ideal speculation level in data security, one can reason that the sum a firm spends to ensure data ought to for the most part be just a little portion of the normal misfortune (i.e., the normal estimation of the misfortune coming about because of a digital/data security rupture).

**1.5 ATTACKER MOTIVATION**

Similarly as with physical security, the inspirations for ruptures of PC security differ between assailants. Some are adrenaline junkies or vandals, others are activists or hoodlums searching for monetary profit. State-supported assailants are presently normal and very much resourced, however began with beginners, for example, Markus Hess who hacked for the KGB, as related by Clifford Stoll, in The Cuckoo's Egg.A standard piece of danger displaying for any framework is to distinguish what may persuade an assault on that framework, and who may be propelled to rupture it. The level and detail of safety measures will change contingent upon the framework to be secured. A home PC, bank, and grouped military system confront altogether different dangers, notwithstanding when the hidden innovations being used are comparable.

**1.6 COMPUTER SECURITY (COUNTER MEASURES)**

In PC security, a countermeasure is an activity, gadget, method, or procedure that decreases a danger, a helplessness, or an assault by dispensing with or counteracting it, by limiting the damage it can cause, or by finding and revealing it so remedial move can be made. Some regular countermeasures are recorded in the accompanying segments:

**1.6.1 Security measures**

A condition of PC "security" is the calculated perfect, achieved utilizing the three procedures: danger avoidance, discovery, and reaction. These procedures depend on different strategies and framework parts, which incorporate the accompanying: User account get to controls and cryptography can ensure frameworks records and information, separately. Firewalls are by a wide margin the most well-known anticipation frameworks from a system security point of view as they can (if appropriately designed) shield access to inner system administrations, and piece certain sorts of assaults through bundle sifting. Firewalls can be both equipment or programming based.

Interruption Detection System (IDS) items are intended to recognize arrange assaults in-advance and aid post-assault crime scene investigation, while review trails and logs serve a comparable capacity for singular frameworks. "Reaction" is essentially characterized by the surveyed security prerequisites of an individual framework and may cover the range from basic overhaul of insurances to notice of legitimate specialists, counter-assaults, and so forth. In some uncommon cases, a total demolition of the traded off framework is favored, as it might happen that not all the bargained assets are recognized.

Today, PC security involves mostly "preventive" measures, similar to firewalls or a leave strategy. A firewall can be characterized as a method for sifting system information between a host or a system and another system, for example, the Internet, and can be actualized as programming running on the machine, guiding into the system stack (or, on account of most UNIX-based working frameworks, for example, Linux, incorporated with the working framework portion) to give constant separating and blocking. Another execution is a supposed physical firewall which comprises of a different machine sifting system movement. Firewalls are normal among machines that are for all time associated with the Internet.

Be that as it may, moderately couple of associations keep up PC frameworks with compelling discovery frameworks, less still have sorted out reaction instruments set up. As result, as Reuters brings up: "Organizations out of the blue report they are losing more through electronic robbery of information than physical taking of advantages". The essential hindrance to compelling destruction of cybercrime could be followed to unnecessary dependence on firewalls and other robotized "location" frameworks. However it is fundamental confirmation assembling by utilizing bundle catch machines that puts crooks in a correctional facility.

The aim of this paper provides E-STAR used for establishing stable and reliable routes in heterogeneous multi-hop wireless networks. The payment system in E-STAR used to reward the nodes which relay others’ packets and charges those that send packets from source to destination. The trust values depends on nodes’ public-key certificates and then develop two routing protocols to direct communicate between them and then highly-trusted nodes having sufficient energy to reduce the probability of breaking the route. E-STAR can stimulate the nodes not only to relay packets, but also to maintain route stability. Experimental results show that E-STAR can secure the payment and trust calculation without error. Simulation results show that routing protocols can improve the packet delivery ratio and route stability.

**CHAPTER 2**

**LITERA­­­TURE SURVEY­­­**

M. Mahmoud et.al. [1] Provides E-STAR used for establishing stable and reliable routes in heterogeneous multi-hop wireless networks. The payment system in E-STAR used to reward the nodes which relay others’ packets and charges those that send packets from source to destination. The trust values depends on nodes’ public-key certificates and then develop two routing protocols to direct communicate between them and then highly-trusted nodes having sufficient energy to reduce the probability of breaking the route. E-STAR can stimulate the nodes not only to relay packets, but also to maintain route stability. Experimental results show that E-STAR can secure the payment and trust calculation without error. Simulation results show that routing protocols can improve the packet delivery ratio and route stability.

Thrasyvoulos Spyropoulos et.al. [2] Communication networks, whether they are wired or wireless, have traditionally been assumed to be connected at least most of the time. However, emerging applications such as emergency response, special operations, smart environments, VANETs, etc. coupled with node heterogeneity and volatile links (e.g. due to wireless propagation phenomena and node mobility) will likely change the typical conditions under which networks operate. In fact, in such scenarios, networks may be mostly disconnected, i.e., most of the time, end-to-end paths connecting every node pair do not exist. To cope with frequent, long-lived disconnections, opportunistic routing techniques have been proposed in which, at every hop, a node decides whether it should forward or store-and-carry a message. Despite a growing number of such proposals, there still exists little consensus on the most suitable routing algorithm(s) in this context. One of the reasons is the large diversity of emerging wireless applications and networks exhibiting such “episodic” connectivity. These networks often have very different characteristics and requirements, making it very difficult, if not impossible, to design a routing solution that fits all. In this paper, we first break up existing routing strategies into a small number of common and tunable routing modules (e.g. message replication, coding, etc.), and then show how and when a given routing module should be used, depending on the set of network characteristics exhibited by the wireless application. We further attempt to create a taxonomy for intermittently connected networks. We try to identify generic network characteristics that are relevant to the routing process (e.g., network density, node heterogeneity, mobility patterns) and dissect different “challenged” wireless networks or applications based on these characteristics. Our goal is to identify a set of useful design guidelines that will enable one to choose an appropriate routing protocol for the application or network in hand. Finally, to demonstrate the utility of our approach, we take up some case studies of challenged wireless networks, and validate some of our routing design principles using simulations.

Y. Liu et.al. [3] Sensor web applications such as real-time environmental decision support systems require the use of sensors from multiple heterogeneous sources for purposes beyond the scope of the original sensor design and deployment. In such cyberenvironments, provenance plays a critical role as it enables users to understand, verify, reproduce, and ascertain the quality of derived data products. Such capabilities are yet to be developed in many sensor web enablement (SWE) applications. This paper develops a provenance-aware “Virtual Sensor” system, where a new persistent live “virtual” sensor is re-published in realtime after some model-based computational transformations of the raw sensor data streams. We describe the underlying OPM (Open Provenance Model) API's (Application Programming Interfaces), architecture for provenance capture, creation of the provenance graph and publishing of the provenance-aware virtual sensor where the new virtual sensor time-series data is augmented with OPM-compliant provenance information. A case study on creating real-time provenance-aware virtual rainfall sensors is illustrated. Such a provenance-aware virtual sensor system allows digital preservation and verification of the new virtual sensors.

L. Moreau et.al. [4] Provenance is well understood in the context of art or digital libaries, where it respectively refers to the documented history of an art object, or the documentation of processes in a digital object’s life cycle. Interest for provenance in the “e-science community” [12] is also growing, since provenance is perceived as a crucial component of workflow systems that can help scientists ensure reproducibility of their scientific analyses and processes

P. Buneman et.al. [5] With the proliferation of database views and curated data- bases, the issue of data provenance- where a piece of data came from and the process by which it arrived in the database - is becoming increasingly important, especially in scientific databases where understanding provenance is crucial to the accuracy and currency of data. In this paper we describe an approach to computing provenance when the data of interest has been created by a database query. We adopt a syntactic approach and present results for a general data model that applies to relational databases as well as to hierarchical data such as XML. A novel aspect of our work is a distinction between “why” provenance (refers to the source data that had some influence on the existence of the data) and “where” provenance (refers to the location(s) in the source databases from which the data was extracted).

Ing-Ray Chen et.al. [6] We propose and analyze a class of trust management protocols for encounter-based routing in delay tolerant networks (DTNs). The underlying idea is to incorporate trust evaluation in the routing protocol, considering not only quality-of-service (QoS) trust properties (connectivity) but also social trust properties (honesty and unselfishness) to evaluate other nodes encountered. Two versions of trust management protocols are considered: an equal-weight QoS and social trust management protocol (called trust-based routing) and a QoS only trust management protocol (called connectivity-based routing). By utilizing a stochastic Petri net model describing a DTN behavior, we analyze the performance characteristics of these two routing protocols in terms of message delivery ratio, latency, and message overhead. We also perform a comparative performance analysis with epidemic routing for a DTN consisting of heterogeneous mobile nodes with vastly different social and networking behaviors. The results indicate that trust-based routing approaches the ideal performance of epidemic routing in delivery ratio, while connectivity-based routing approaches the ideal performance in message delay of epidemic routing, especially as the percentage of selfish and malicious nodes present in the DTN system increases. By properly selecting weights associated with QoS and social trust metrics for trust evaluation, our trust management protocols can approximate the ideal performance obtainable by epidemic routing in delivery ratio and message delay without incurring high message overhead.

Ing-Ray Chen et.al. [7] Delay tolerant networks (DTNs) are characterized by high end-to-end latency, frequent disconnection, and opportunistic communication over unreliable wireless links. In this paper, we design and validate a dynamic trust management protocol for secure routing optimization in DTN environments in the presence of well-behaved, selfish and malicious nodes. We develop a novel model-based methodology for the analysis of our trust protocol and validate it via extensive simulation. Moreover, we address dynamic trust management, i.e., determining and applying the best operational settings at runtime in response to dynamically changing network conditions to minimize trust bias and to maximize the routing application performance. We perform a comparative analysis of our proposed routing protocol against Bayesian trust-based and non-trust based (PROPHET and epidemic) routing protocols. The results demonstrate that our protocol is able to deal with selfish behaviors and is resilient against trust-related attacks. Furthermore, our trust-based routing protocol can effectively trade off message overhead and message delay for a significant gain in delivery ratio. Our trust-based routing protocol operating under identified best settings outperforms Bayesian trust-based routing and PROPHET, and approaches the ideal performance of epidemic routing in delivery ratio and message delay without incurring high message or protocol maintenance overhead.

Jin-Hee Cho et.al. [8] Managing trust efficiently and effectively is critical to facilitating cooperation or collaboration and decision making tasks in tactical networks while meeting system goals such as reliability, availability, or scalability. Delay tolerant networks are often encountered in military network environments where end-to-end connectivity is not guaranteed due to frequent disconnection or delay. This work proposes a provenance-based trust framework for efficiency in resource consumption as well as effectiveness in trust evaluation. Provenance refers to the history of ownership of a valued object or information. We adopt the concept of provenance in that trustworthiness of an information provider affects that of information, and vice-versa. The proposed trust framework takes a data-driven approach to reduce resource consumption in the presence of selfish or malicious nodes. This work adopts a model-based method to evaluate the proposed trust framework using Stochastic Petri Nets. The results show that the proposed trust framework achieves desirable accuracy of trust evaluation of nodes compared with an existing scheme while consuming significantly less communication overhead.

Juliana Freire et.al. [9] Some important concepts related to systematically capture and manage provenance for computational tasks that has relevance to a wide range of domains and applications have been discussed. A provenance management solution comprises three main components, a capture mechanism, a representational model, and an infrastructure for storage, access, and queries. A provenance capture mechanism requires access to a computational task's relevant details such as its steps, execution information, and user-specific annotations. Operating system (OS)-based capture mechanisms do not require modifications to existing processes and are agnostic about how tasks are modeled. A wide variety of data models and storage systems ranging from specialized Semantic Web languages and XML dialects stored as files to tuples stored in relational database tables have also been proposed to facilitate an infrastructure for effectively and efficiently querying data in a provenance management system.

**CHAPTER 3**

**MOTIVATION AND OBJECTIVES**

# **3 EXISTING SYSTEM**

In Delay Tolerant Networks(DTNs), the peer-to-peer connectivity between every node are not continuous and hence results in loss of information. It allows the system to be susceptible against corruption of data and other similar threats.

Provenance is the chronology of the ownership, custody or location of a historical object. Interest for provenance in the “e-science community” is also growing, since provenance is perceived as a crucial component of workflow systems that can help scientists ensure reproducibility of their scientific analyses and processes.

Open provenance model (OPM) is used in the existing system. The Open Provenance Model (OPM) is a model for provenance that is designed to meet the following requirements:

* To allow provenance information to be exchanged between systems, by means of a compatibility layer based on a shared provenance model.
* To allow developers to build and share tools that operate on such provenance model etc.

It is traced back by the second provenance challenge which aimed at establishing inter-operability of systems, by exchanging provenance information. It has three components called artifacts, processes and agents. Artifacts represent an immutable piece of state, which may have a physical embodiment in a physical object, or a digital representation in a computer system. Processes represent actions performed on or caused by artifacts, and resulting in new artifacts. Agents represent contextual entities acting as a catalyst of a process, enabling, facilitating, controlling, or affecting its execution. It records the past execution of the three components.

**3.1 Problems in Existing System:**

1. Open provenance model (OPM) does not prove the security of the data.
2. Provenance verification is not accurate in the existing system.
3. It is critical to improve the security in the open provenance model.
4. It is not suitable for the end to end connectivity in the delay tolerant network.

# **3.2 Proposed System**

In this project, to solve the problem of lack of data provenance provided by the OPM, a better model viz, the Provest model i.e., Provenance Based Trust Model is used. This model provides a better data provenance which is very important in the delay tolerant network.

The lack of direct interaction experience in DTN environments hinders continuous evidence collection and can result in incorrect trust estimation, leading to poor application performance. In this model, the use of provenance information for evidence propagation for sparse DTNs without solely relying on encounter-based evidence exchange is achieved.

Data are transferred in the network by the nodes present. Here a cluster or an area of nodes including both the source and destination forms a sub network. This makes it difficult for foreign nodes to disrupt the data flow between the source and destination.

Provenance based trust model (PROVEST) is used to check the provenance of the data. Data is transferred from the source to destination with the security. Provenance is verified by the destination at last in the model.

**3.2.1 Advantages:**

1. Security of the data is increased.
2. Destination can easily verify the provenance of the data.
3. And the destination can check the intermediate nodes that transfer the data.
4. Malfunction of the intermediates are avoided by this mechanism.

**3.3 OUTLOOK OF AT PROPOSED SYSTEM**

* The user/source who is the data owner uses the proposed system to upload files and perform other file management operation.
* The proposed system also allows the user to select the number of nodes to be used during data transmission between them including the source and destination.
* The proposed system then makes use of AES and Hashing algorithm for encryption, thus enhancing the security of the data.

Further, we can find the system consists of 4 modules, which are:

* Key Parameters Distribution Module
* Key and Hash Generation Module
* Trust Aggregation Module
* Trust Verification Module

**KEY PARAMETERS DISTRIBUTION MODULE**

In this module, the network with nodes are created. All the nodes would have the ids and energy. Trusted authority will generate the key parameters for the members in the network. The key parameters are distributed to the nodes in the network. Using this key parameter, the keys are generated.

**KEY AND HASH GENERATION MODULE**

In this module all the sources, intermediates and destinations generate the keys. Quana1.4 is a tool used to generate the keys by the parameters. Using that keys, the data is encrypted. Then the hash for the ids are generated. These are used for the provenance. Provenance information are verified by the destination.

**TRUST AGGREGATION MODULE**

This module contains destination which has the encrypted data and the hash keys. Destination decrypts the data by using the group key which is already encrypted by the source by the group key. All the hashes from all the nodes in the network are received in the destination. It is called as the trust aggregation.

**TRUST VERIFICATION MODULE**

In this module, the hashes generated by the nodes in the network are verified by the destination. Keys generated by the qana1.4 is used for the hash verification. Trust verification is used to make decision about the trust of the data. If any one of the hash is changed means the destination does not trust the data. Security is also high when compared with the existing mechanism.

**CHAPTER 4**

**SYSTEM REQUIREMENTS SPECIFICATION AND DESIGN**

**4.1 SOFTWARE REQUIREMENT SPECIFICATION**

**4.1.1 Non-Functional Requirements**

The non-functional requirements include time constraints and constraints on the development process and standards. The non-functional requirements are as follows:

* Speed: The given input should be processed into the correct output within appropriate time.
* Ease of use: The system should be user friendly for the customers to use, so it doesn’t require much training time.
* Reliability: The failure rate of the system should be low, then it is accepted as reliable.
* Portability: The system should be easily implemented.

**4.1.2 Functional Requirements**

The functional requirements are:

* User Interfaces: The external users are the clients. All the clients can use this software for indexing and searching.
* Hardware Interfaces: The external hardware interface used for indexing and searching is personal computers of the clients. The PC’s may be laptops with wireless LAN as the internet connections provided will be wireless.
* Software Interfaces: The Operating Systems can be any version of Windows.
* Performance Requirements: The PC’s used must be at least Pentium 4 machines so that they can give optimum performance of the product.

# **4.1.3 Minimum Sytem Requirements**

**4.1.3.1 Hardware Requirements:**

* Processor : Pentium IV
* System : Core 2 duo
* Speed : 2.8 GHz
* RAM : 1 GB
* Hard Disk : 20 GB

**4.1.3.2 Software Requirements:**

* Operating System : Win XP/7/8/10
* Language : Java
* I D E : Eclipse
* Tool : Qana 1.4

**4.2 HIGH LEVEL DESIGN**

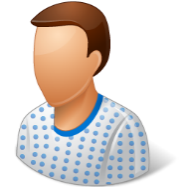
**4.2.1 System Architecture**



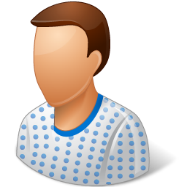
Trusted Authority



Source



Intermediate1



Intermediate2



Destination



Trusted Authority

Encrypt Data and Hash ID

Hash ID

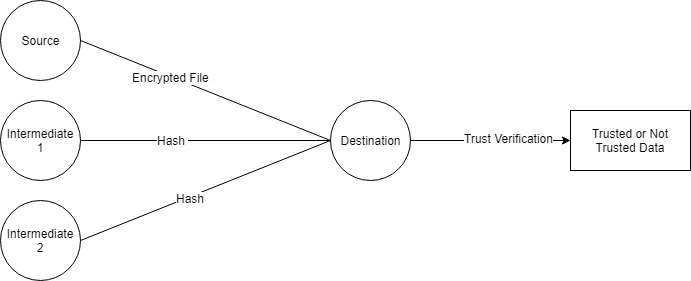
Hash ID

Trust Aggregation and Verification

Fig 5.1: System Architecture

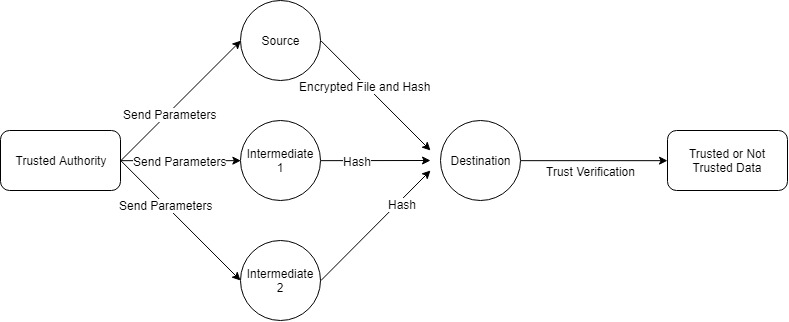
The above figure represents the various modules of the proposed system and how they interrelate with each other. Source encrypts and file and sends it to the destination. Two intermediaries that are part of the Delay Tolerant Network and are closer to source are selected. These intermediaries creates hash of their ID’s and send it to destination. On receiving the file, Destination first decrypts the file and verifies the trust depending on the hashed values of the source and intermediaries.

* + 1. **Data Flow Diagrams**



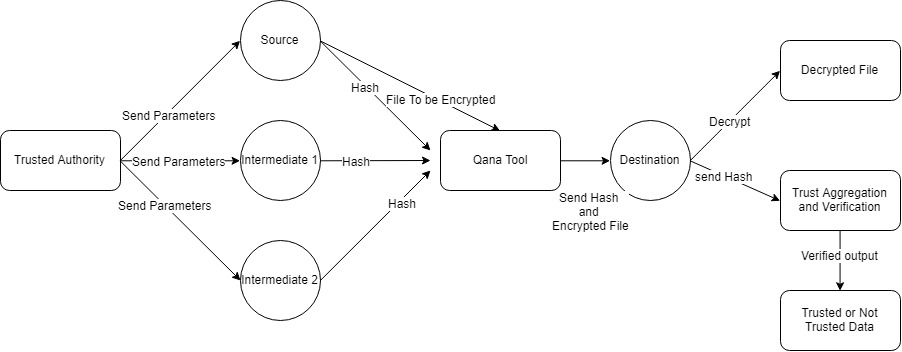
**Figure 4.3 Data Flow Diagram Level 0**

DFD 0: This Diagram shows the Source, Intermediate 1 and Intermediate 2 sending the Hash of their Id’s to the Destination. On receiving these hash, the destination first will decrypt the data and then says if the Data is trusted or not



**Figure 4.2.4 Data Flow Diagram Level 1**

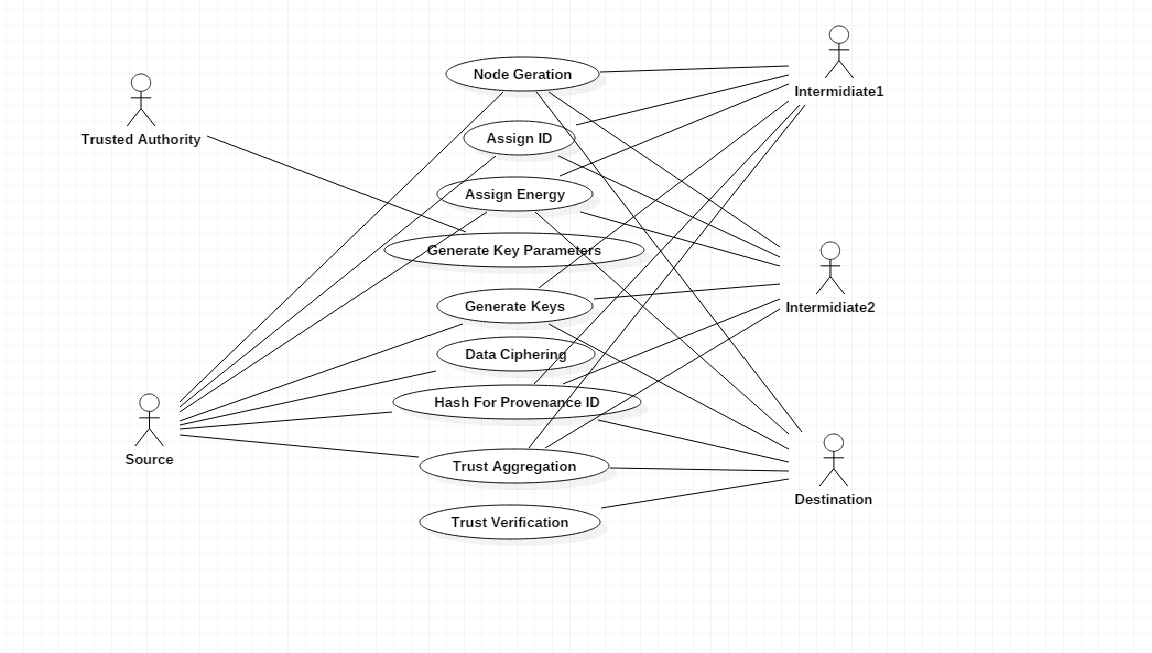
DFD 1 : This diagram shows the Trusted Authority coming into picture. It randomly generates parameters to the source and intermediaries. Now source receives the parameters and using these parameters it creates a group key and uses it to encrypt an input file as well as creating hash of its ID. In the Same manner, Both intermediaries uses the received parameters to create hash of their IDs. These hash are used by destination to say the output.



**Figure 4.2.4 Data Flow Diagram Level 2**

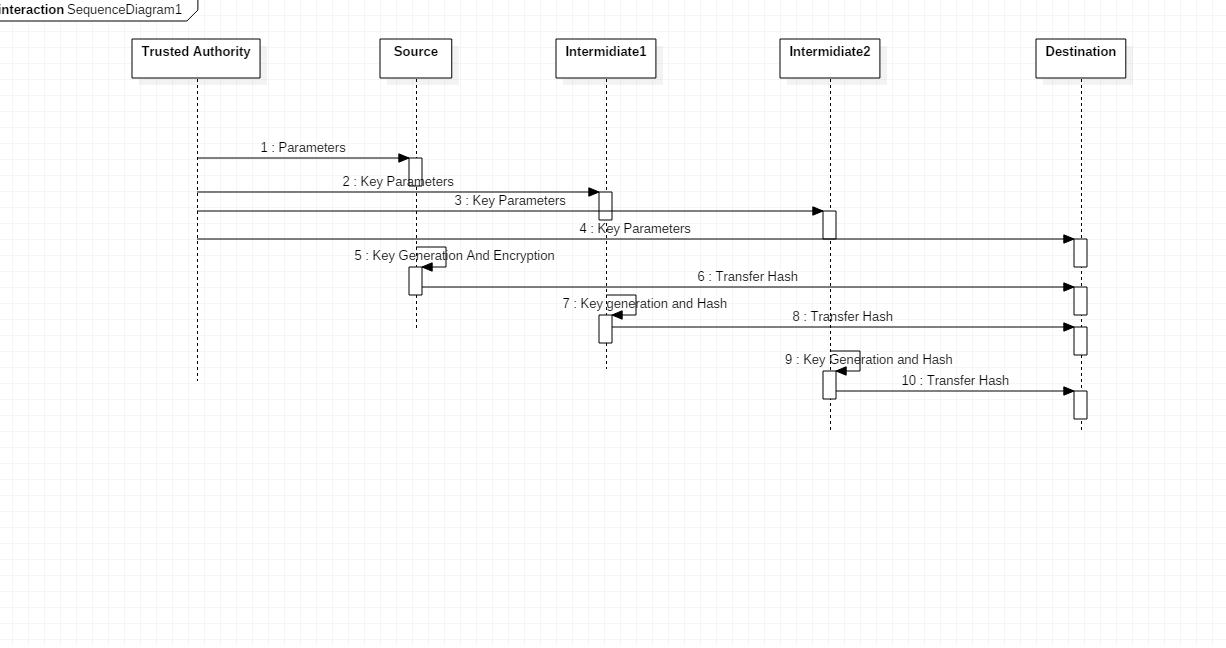
DFD 2 : Trusted Authority randomly generates the parameters and sends it to source and intermediaries. Using these parameters source and intermediaries creates their ID’s hash. These hash are created using qana tool which requires it to have a group key that is shared by all the nodes. Source encrypts the input file using these parameters. The Hashed archives are stored locally and in destination, these archives are extracted. These extracted ID’s are compared with the verification ID’s which are the ID’s created and stored in a text file when nodes are created. If the Trust ids match with all the created node ids the destination decides the decrypted file must be considered and outputs it as Trusted Data and Not Trusted Data Otherwise.

**4.2.3 Use Case Diagram**

 **Figure 4.6 Use Case Diagram**

Use case diagram at its simplest is a representation of user’s interaction with the system. Use case diagram shows various users of a system, relationships among them and their use cases. In figure 6.1, various users of the system are Source and Destination, various use cases are Node Generation, Generate Parameters, Assign Id, Generate key, Data ciphering, Trust aggregation, Trust verification etc.

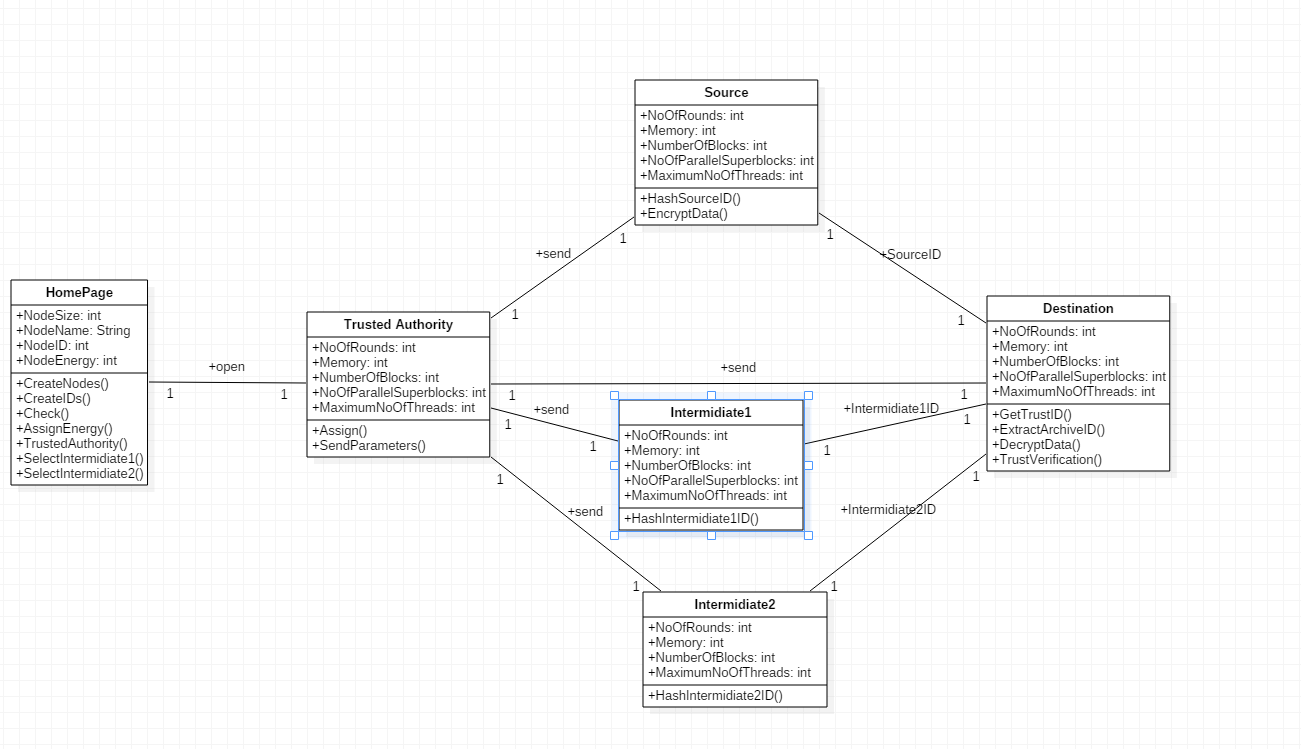
**4.2.4 Sequence Diagram**

****

**Figure 4.6 Sequence Diagram**

**4.2.5 Class Diagram**

Class diagram is an illustration of the relationships and [source code](https://searchmicroservices.techtarget.com/definition/source-code) dependencies among classes in the Unified Modeling Language (UML). In this context, a [class](https://whatis.techtarget.com/definition/class) defines the [method](https://whatis.techtarget.com/definition/method)s and [variable](https://whatis.techtarget.com/definition/variable)s in an [object](https://searchmicroservices.techtarget.com/definition/object), which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP). The concept is several years old but has been refined as OOP modeling paradigms have evolved.



**Figure 4.7 Class diagram**

Home Page: This class declares the nodes, its ids, and energy. It also assigns selected the source, destination and the intermediaries. If source and destination are same then it provides a message pop up to select different nodes for source and destination. Intermediaries are selected randomly from the Node ID’s other than the ones selected for source and destination.

Trusted Authority: This creates and assigns parameters for global key that is used to create the hash. These parameters are randomly created for each of the nodes and sent in a socket.

Source : Takes the parameters as input and uses it in global key to encrypt a input file that is sent to destination, later it creates a hash of its own ID.

Intermediaries : They receive the parameters from trusted authority and uses these to create the hash of their own ids.

Destination : This node first decrypts the input file using a global key and extracts the contents in that was in the input file. It uses hash verification function to extract the hash of all the nodes and calculates the trust from the extracted node ids comparing with the initialsed node ids

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 PROGRAMMING LANGUAGE**

**5.1.1 Java**

Java is an arrangement of computer software and specifications created by Sun Microsystems, which was later procured by the Oracle Corporation, which gives a framework to creating application programming and sending it in a crossplatform figuring condition. Java is utilized as a part of a wide assortment of processing stages from implanted gadgets and cell phones to big business servers and supercomputers. Writing in the Java programming dialect is the essential method to deliver code that will be sent as byte code in a Java Virtual Machine (JVM).

The Java stage is a suite of projects that encourage creating and running projects written in the Java programming dialect. A Java stage will incorporate an execution motor (called a virtual machine), a compiler and an arrangement of libraries; there may likewise be extra servers and elective libraries that rely upon the necessities. Java isn't particular to any processor or working framework as Java stages have been executed for a wide assortment of equipment and working frameworks with a view to empower Java projects to run indistinguishably on every one of them.

The Java Development Kit (JDK) is a Sun item went for Java engineers. Since the presentation of Java, it has been by a wide margin the most generally utilized Java Software Development Kit (SDK). It contains a Java compiler, a full duplicate of the Java Runtime Environment (JRE), and numerous other vital advancement instruments.

**5.2 J2EE**

A Java EE application or a Java Platform, Enterprise Edition application is any deployable unit of Java EE usefulness. This can be a solitary Java EE module or a gathering of modules bundled into an EAR document alongside a Java EE application organization descriptor. Java EE applications are regularly built to be disseminated over numerous registering levels.

* Venture applications can comprise of the accompanying:
* EJB modules (bundled in JAR documents);
* Web modules (bundled in WAR documents);
* Connector modules or asset connectors (bundled in RAR records);
* Session Initiation Protocol (SIP) modules (bundled in SAR records);
* Application customer modules;
* Additional JAR records containing subordinate classes or different segments required by the application;

Java Platform, Enterprise Edition or Java EE is Oracle's venture java processing stage. The stage gives an API and runtime condition for creating and running undertaking programming, including system and web administrations, and other vast scale, multi-layered, versatile, dependable, and secure system applications. Java EE broadens the Java Platform, Standard Edition (Java SE), giving an API to adaptation to non-critical failure, protest social mapping, dispersed and multi-level designs, and web administrations. The stage fuses an outline construct to a great extent in light of measured segments running on an application server. Programming for Java EE is essentially created in the Java programming dialect and utilizations XML for design.

**5.3 Eclipse**

Eclipse is a stage that has been outlined starting from the earliest stage for building incorporated web and application advancement tooling. By outline, the stage does not give a lot of end client usefulness independent from anyone else. The estimation of the stage is the thing that it energizes: fast advancement of coordinated highlights in view of a module show.

Overshadowing gives a typical (UI) demonstrate for working with instruments. It is intended to keep running on different working frameworks while furnishing hearty combination with each fundamental OS. Modules can program to the Eclipse versatile APIs and run unaltered on any of the bolstered working frameworks.

At the center of Eclipse is a design for dynamic disclosure, stacking, and running of modules. The stage handles the coordinations of finding and running the correct code. The stage UI gives a standard client route show. Each module would then be able to center around completing few undertakings well.

**Key Parameters Distribution:**

* First of all create the network with nodes.
* All the nodes have the ids and energy.
* Trusted authority will generate the key parameters for the members in the network.
* The key parameters are distributed to the nodes in the network.
* Using this key parameters the keys are generated.

**Key and Hash Generation:**

* All the sources, intermediates and destinations generate the keys.
* Quana1.4 is a tool used to generate the keys by the parameters.
* Using that keys the data are encrypted.
* Then the hash for the id’s are generated.
* These are used for the provenance. Provenance informations are verified by the destination.

**Trust Aggregation:**

* Destination having the encrypted data and the hash keys.
* Destination decrypts the data by using the group key.
* This data is already encrypted by the source by the group key.
* All the hashes from all the nodes in the network are received in the destination.
* It is called as the trust aggregation.

**Trust Verification:**

* The hashes generated by the nodes in the network are verified by the destination.
* Keys generated by the qana1.4 is used for the hash verification.
* Trust verification is used to make decision about the trust of the data.
* If any one of the hash is changed means the destination does not trust the data.
* Security is also high when compared with the existing mechanism.

**Working:** Open the homepage of the provenance model. A GUI window will open. Select the no of nodes, for example five in the spinner. Then on clicking Create Nodes, it gets the node size form the spinner and appends it to the Text Field Below it. On clicking Create ID, it loops through the size of the nodes, and for 5 iterations, it assigns an Id to each Node, therefore in this case five Id’s in total. Now it also assigns the ID’s to the comboBox for source and destination on the right. On clicking Assign Energy, it assigns energy by looping over the node size, taking the random of that no. and added 1000 to it. A node is assigned an initialenergy level selected from the range of [e; e0] (in hrs:) and its energy consumption is affected by its availability to serve requests. On selecting “Select Intermediate 1”, it checks whether the source Id and the destination Id are not equal to the Intermidiate 1 Id which is assigned randomly. Now if they are not equal, then using FileWriter within BufferedWriter, we write the Id for the intermediate inside a text file. On clicking “ Select Intermidiate 2”, we check if the source, destination and the Intermidiate1 id’s assigned previously are not equal to Intermidiate 2 Id. If so we erite the intrermidiate Id to a text File.

On selecting “Trusted Authority”, we set the GUI window of trusted Authority in the middle of the screen and set visible to true. Inside trusted authority we assign Id’s for each of the parameters by randomizing and we send those parameters to source, destination, Intermidiate1 and Intermidiate2 respectively using a Server Socket.

In the Source , after receiving the parameters , encrypt a text file which is to be verified using Qana, a tool in Java through symmetric key encryption using the cryptographic algorithm called Scrypt. After that we display the encrypted text inside the encrypted data text area. Similarly we display the hash id data within the hash Id text area.

In the Intermidiates 1 and 2, we receive the parameters from trusted authority and then display the hash Id inside the hash id text Area.

Inside The destination we, decrypt the data after receiving the parameters from trusted authority and generate trust id’s for the text file. 3 trust Ids ,source, Intermidiate 1 and Intermidiate 2. These trust ID’s are verified with the original Ids for each of them and we iterate 1 every time the trust id matches with each of the ids. If all three of them matches then on click of trust verification, we get “Trusted Data” ad the trust level is 3. Else we get the message saying “ Not Trusted Data”.

**Code:**

**Homepage.java**

**private** **void** jButton1ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

jTextArea1.append("Sl No\tNodes\n");

jTextArea1.append("-------\t----------\n");

non = Integer.*parseInt*(jSpinner1.getValue().toString());

**for** (**int** i = 0; i < non; i++) {

jTextArea1.append((i + 1) + "\tNode " + (i + 1) + "\n");

*nodeNames*.add("Node " + (i + 1));

}

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton2ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

jTextArea2.append("Sl No\tNode Names\tNode Ids\n");

jTextArea2.append("-------\t-------------------\t--------------\n");

**for** (**int** i = 0; i < *nodeNames*.size(); i++) {

String nodeIds = Integer.*toString*(Math.*abs*(r.nextInt()));

**this**.*nodeIds*.add(nodeIds);

jComboBox1.addItem(nodeIds);

jComboBox2.addItem(nodeIds);

jTextArea2.append((i + 1) + "\t" + *nodeNames*.get(i) + "\t" + nodeIds + "\n");

}

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jComboBox1ItemStateChanged(java.awt.event.ItemEvent evt)

**try** {

*source* = jComboBox1.getSelectedItem().toString();

jTextField1.setText(*source*);

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jComboBox2ItemStateChanged(java.awt.event.ItemEvent evt)

**try** {

*destination* = jComboBox2.getSelectedItem().toString();

jTextField2.setText(*destination*);

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton3ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

**if** (*source*.equals(*destination*)) {

JOptionPane.*showMessageDialog*(**null**, "Select Different Nodes");

} **else** {

jComboBox1.setEnabled(**false**);

jComboBox2.setEnabled(**false**);

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("C:\\provest code\\code\\provest\\Source\\SourceId.txt"));

bw.write(*source*);

bw.close();

}

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** formWindowOpened(java.awt.event.WindowEvent evt)

**try** {

jTextField1.setEditable(**false**);

jTextField2.setEditable(**false**);

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton4ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

jTextArea3.append("Node Id\tNode Names\tNode Energy\n");

jTextArea3.append("-----------\t-------------------\t------------------\n");

**for** (**int** i = 0; i < *nodeNames*.size(); i++) {

String nodeEnergy = Integer.*toString*(Math.*abs*(r.nextInt(2000) + 1000));

**this**.*nodeEnergy*.add(nodeEnergy);

jTextArea3.append(*nodeIds*.get(i) + "\t" + *nodeNames*.get(i) + "\t" + nodeEnergy + "\n");

}

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton8ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

trustedauthority ta = **new** trustedauthority();

ta.setLocationRelativeTo(**null**);

ta.setVisible(**true**);

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton5ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

**while** (**true**) {

**int** ran = r.nextInt(*nodeIds*.size());

*intermediate1* = *nodeIds*.get(ran);

**if** ((!*source*.equals(*intermediate1*)) && (!*destination*.equals(*intermediate1*))) {

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("C:\\provest code\\code\\provest\\Intermediate1\\Intermediate1Id.txt"));

bw.write(*intermediate1*);

bw.close();

**break**;

}

}

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton6ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

**while** (**true**) {

**int** ran = r.nextInt(*nodeIds*.size());

*intermediate2* = *nodeIds*.get(ran);

**if** ((!*source*.equals(*intermediate2*)) && (!*destination*.equals(*intermediate2*))&& (!*intermediate1*.equals(*intermediate2*))) {

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("C:\\provest code\\code\\provest\\Intermediate2\\Intermediate1Id.txt"));

bw.write(*intermediate2*);

bw.close();

**break**;

}

}

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**Source.java**

**private** **void** jButton2ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

ProcessBuilder pb = **new** ProcessBuilder("java", "-jar", "./qana.jar");

Process p = pb.start();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton3ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

String filePath = **null**;

JFileChooser fc=**new** JFileChooser();

**int** a=fc.showOpenDialog(**null**);

**if**(a==JFileChooser.***APPROVE\_OPTION***)

{

File fileToOpen=fc.getSelectedFile();

filePath=fileToOpen.toString();

}

BufferedReader br=**new** BufferedReader(**new** FileReader(filePath));

String content;

**while**((content=br.readLine())!=**null**){

jTextArea1.append(content+"\n");

}

br.close();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton4ActionPerformed(java.awt.event.ActionEvent evt)

**try**{

String filePath = **null**;

JFileChooser fc=**new** JFileChooser();

**int** a=fc.showOpenDialog(**null**);

**if**(a==JFileChooser.***APPROVE\_OPTION***)

{

File fileToOpen=fc.getSelectedFile();

filePath=fileToOpen.toString();

}

BufferedReader br=**new** BufferedReader(**new** FileReader(filePath));

String content;

**while**((content=br.readLine())!=**null**){

jTextArea2.append(content+"\n");

}

br.close();

}

**catch**(Exception e){

System.***out***.println(e.getMessage());

}

}

**public** **static** **void** main(String args[]) {

**try** {

**for** (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.*getInstalledLookAndFeels*()) {

**if** ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.*setLookAndFeel*(info.getClassName());

**break**;

}

}

} **catch** (ClassNotFoundException ex) {

java.util.logging.Logger.*getLogger*(source.**class**.getName()).log(java.util.logging.Level.***SEVERE***, **null**, ex);

} **catch** (InstantiationException ex) {

java.util.logging.Logger.*getLogger*(source.**class**.getName()).log(java.util.logging.Level.***SEVERE***, **null**, ex);

} **catch** (IllegalAccessException ex) {

java.util.logging.Logger.*getLogger*(source.**class**.getName()).log(java.util.logging.Level.***SEVERE***, **null**, ex);

} **catch** (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.*getLogger*(source.**class**.getName()).log(java.util.logging.Level.***SEVERE***, **null**, ex);

}

java.awt.EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**new** source().setVisible(**true**);

}

});

}

**Intermidiates.java**

**private** **void** jButton4ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

ProcessBuilder pb = **new** ProcessBuilder("java", "-jar", "./qana.jar");

Process p = pb.start();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton5ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

String filePath = **null**;

JFileChooser fc = **new** JFileChooser();

**int** a = fc.showOpenDialog(**null**);

**if** (a == JFileChooser.***APPROVE\_OPTION***) {

File fileToOpen = fc.getSelectedFile();

filePath = fileToOpen.toString();

}

BufferedReader br = **new** BufferedReader(**new** FileReader(filePath));

String content;

**while** ((content = br.readLine()) != **null**) {

jTextArea2.append(content + "\n");

}

br.close();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton6ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

String filePath = **null**;

JFileChooser fc = **new** JFileChooser();

**int** a = fc.showOpenDialog(**null**);

**if** (a == JFileChooser.***APPROVE\_OPTION***) {

File fileToOpen = fc.getSelectedFile();

filePath = fileToOpen.toString();

}

BufferedReader br = **new** BufferedReader(**new** FileReader(filePath));

String content;

**while** ((content = br.readLine()) != **null**) {

jTextArea3.append(content + "\n");

}

br.close();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** formWindowOpened(java.awt.event.WindowEvent evt)

**try** {

BufferedReader br=**new** BufferedReader(**new** FileReader("C:\\provest code\\code\\provest\\Intermediate1\\Intermediate1Id.txt"));

String intermediate1=br.readLine();

br.close();

jLabel3.setText("Id " + intermediate1);

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

**try** {

BufferedReader br=**new** BufferedReader(**new** FileReader("C:\\provest code\\code\\provest\\Intermediate2\\Intermediate2Id.txt"));

String intermediate2=br.readLine();

br.close();

jLabel14.setText("Id " + intermediate2);

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**Destination.java**

**private** **void** jButton1ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

Socket s = **new** Socket("localhost", 1234);

BufferedReader br = **new** BufferedReader(**new** InputStreamReader(s.getInputStream()));

jTextField1.setText(br.readLine());

jTextField2.setText(br.readLine());

jTextField3.setText(br.readLine());

jTextField4.setText(br.readLine());

jTextField5.setText(br.readLine());

br.close();

s.close();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton2ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

ProcessBuilder pb = **new** ProcessBuilder("java", "-jar", "./qana.jar");

Process p = pb.start();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton3ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

ProcessBuilder pb = **new** ProcessBuilder("java", "-jar", "./qana.jar");

Process p = pb.start();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

**try** {

String filePath = **null**;

JFileChooser fc = **new** JFileChooser();

**int** a = fc.showOpenDialog(**null**);

**if** (a == JFileChooser.***APPROVE\_OPTION***) {

File fileToOpen = fc.getSelectedFile();

filePath = fileToOpen.toString();

}

BufferedReader br = **new** BufferedReader(**new** FileReader(filePath));

String content;

**while** ((content = br.readLine()) != **null**) {

jTextArea1.append(content + "\n");

}

br.close();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton4ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

BufferedReader br=**new** BufferedReader(**new** FileReader("C:\\provest code\\code\\provest\\Source\\SourceId.txt"));

String intermediate1=br.readLine();

br.close();

jTextField6.setText(intermediate1);

*trustId*.add(intermediate1);

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

**try** {

BufferedReader br=**new** BufferedReader(**new** FileReader("C:\\provest code\\code\\provest\\Intermediate1\\Intermediate1Id.txt"));

String intermediate1=br.readLine();

br.close();

jTextField7.setText(intermediate1);

*trustId*.add(intermediate1);

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

**try** {

BufferedReader br=**new** BufferedReader(**new** FileReader("C:\\provest code\\code\\provest\\Intermediate2\\Intermediate2Id.txt"));

String intermediate2=br.readLine();

br.close();

jTextField8.setText(intermediate2);

*trustId*.add(intermediate2);

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton5ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

ProcessBuilder pb = **new** ProcessBuilder("java", "-jar", "./qana.jar");

Process p = pb.start();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**private** **void** jButton6ActionPerformed(java.awt.event.ActionEvent evt)

**try** {

BufferedReader br=**new** BufferedReader(**new** FileReader("C:\\provest code\\code\\provest\\Destination\\Source\\SourceId.txt"));

String intermediate1=br.readLine();

br.close();

jTextArea2.append(intermediate1+"\n");

*verificationId*.add(intermediate1);

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

**try** {

BufferedReader br=**new** BufferedReader(**new** FileReader("C:\\provest code\\code\\provest\\Destination\\Intermediate1\\Intermediate1Id.txt"));

String intermediate1=br.readLine();

br.close();

jTextArea2.append(intermediate1+"\n");

*verificationId*.add(intermediate1);

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

**try** {

BufferedReader br=**new** BufferedReader(**new** FileReader("C:\\provest code\\code\\provest\\Destination\\Intermediate2\\Intermediate2Id.txt"));

String intermediate2=br.readLine();

br.close();

jTextArea2.append(intermediate2+"\n");

*verificationId*.add(intermediate2);

**int** trust=0;

**for**(**int** i=0;i<*verificationId*.size();i++){

**if**(*trustId*.contains(*verificationId*.get(i))){

trust++;

}

}

**if**(trust==3){

JOptionPane.*showMessageDialog*(**null**, "Trusted Data");

}

**else**{

JOptionPane.*showMessageDialog*(**null**, "Not Trusted Data");

}

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**CHAPTER 6**

**RESULTS AND DISCUSSIONS**

**6.1 TESTING**

**6.1.1 Unit Testing**

The basic level of testing is called unit testing. This checks if the all the small modules of the software are working fine. The specific modules are checked against the design specifications that are formulated during the design phase .The internal working of the software is checked.The programming of the specific modules and unit testing are usually done parallely.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Objective** | **Input** | **Expected Results** | **Success/Failure** |
| TC\_001 | Load the document | Document | The document should be fetched without any errors | Success |
| TC\_002 | Keyword Extraction | The fetched document | Extracting most attentive keywords | Success |
| TC\_003 | Marking data | Document with extracted keywords | The keywords will be highlighted in yellow | Success |
| TC\_004 | Server | Extracted keywords | Receive the keywords from server | Success |

**Table 6.1 Unit Testing**

**6.1.2 Integration Testing**

Integrated testing is the next step after unit testing in software testing where the individual components are combined in multiple ways and tested.

The different modules in this are the client and server. These two modules should be tested after integration. The data that is sent by the client should be received by the server. The first level of summarized data should be sent to the server. The server should receive this and mark any new words that were not already marked at the client side.

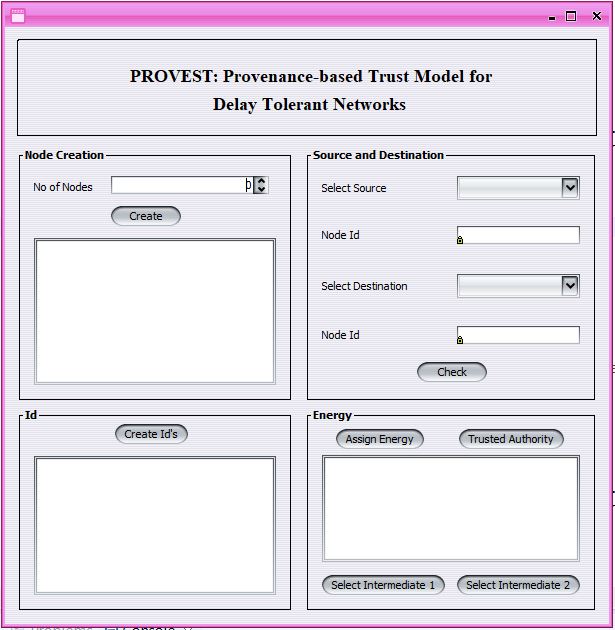
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Objective** | **Input** | **Expected Results** | **Success/Failure** |
| TC\_001 | Check if servers are  running properly | Data is passed from client to server | Data is received at server | Success |
| TC\_002 | Check if servers are  running properly | Data is passed from client to server | Data is received at server | Failure  (If server is not started before running client) |
| TC\_003 | Marking keywords | First level summarized sentences | Keywords in the semantic dictionary should be marked | Success |

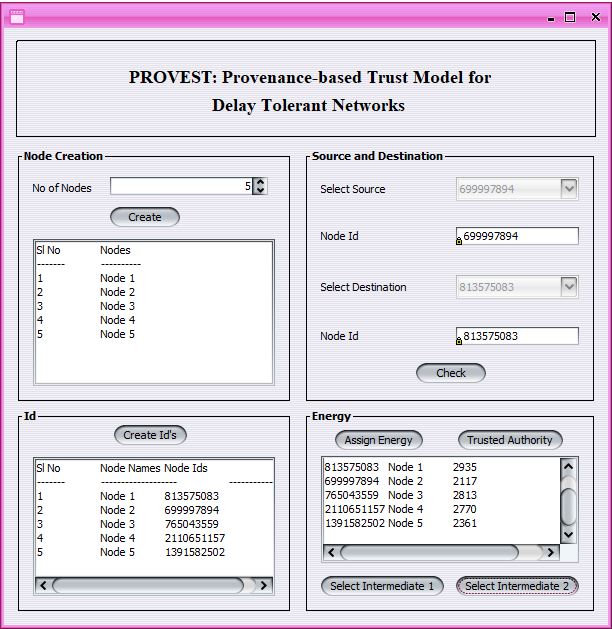
**Table 6.2 Integration Testing**

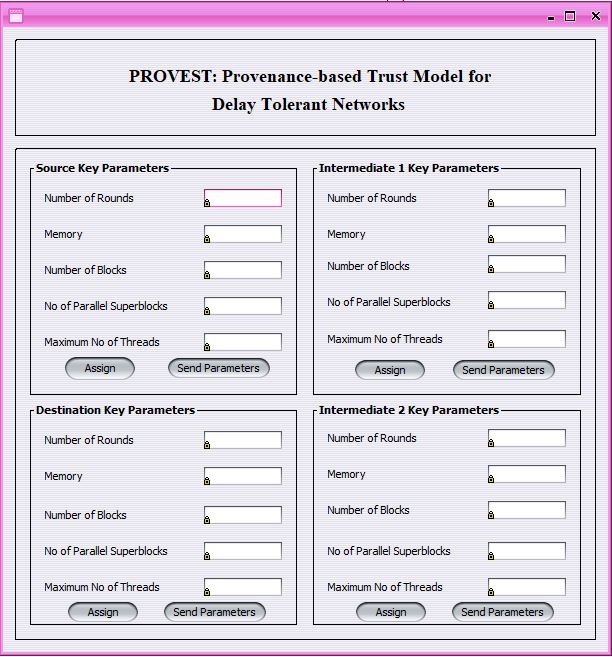
**6.1.3 System Testing**

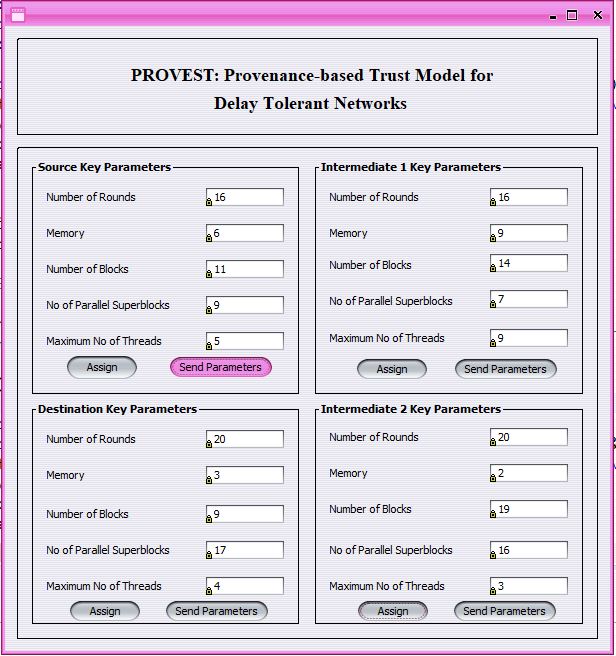
In system testing, the overall working of the system is being tested. All the integrated components are working fine together and are producing the required output. The system has a graphical user interface where the user can select the document . User has to click on the various buttons like checking which cluster the document belongs to . The user has to then click on the other buttons such as extract , mark which performs the particular functions . Then the data is sent to the server which performs the functions such as checking with the semantic dictionary and marking those words in the document data. Since the user interface is simple it would be very easy for the user to interact with the system.

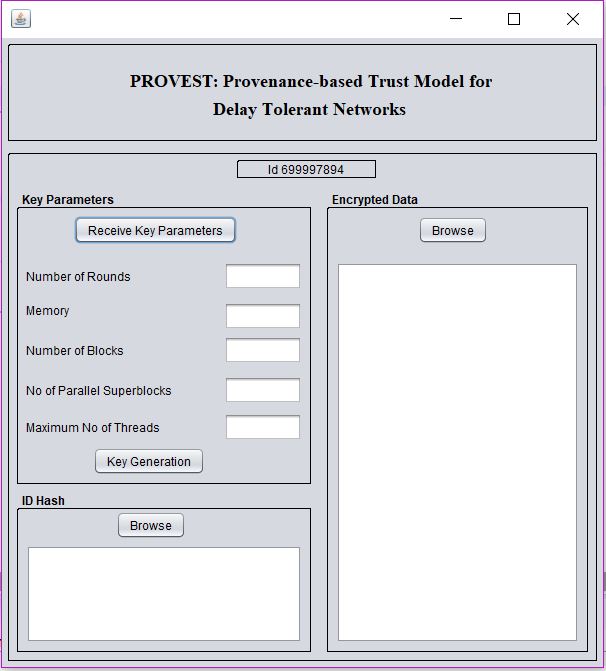
**6.2 Snapshots and Discussions**

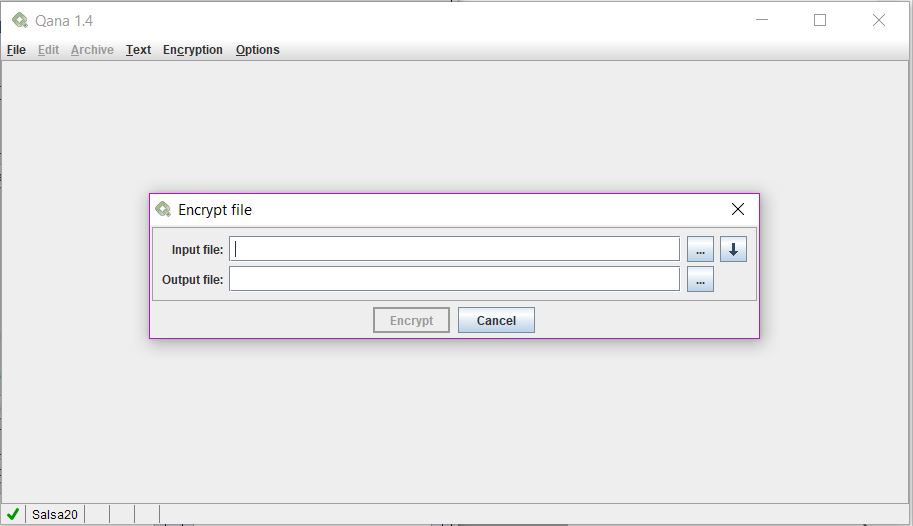


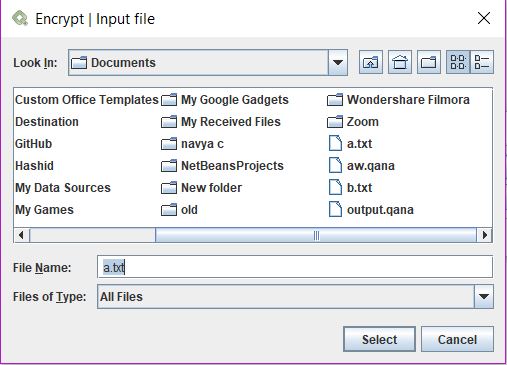


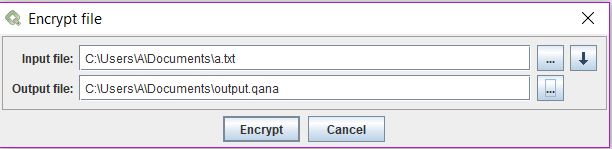


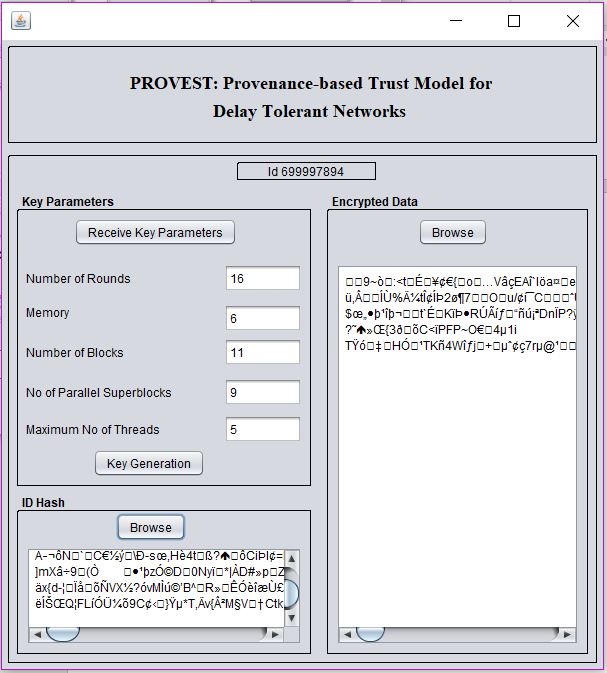


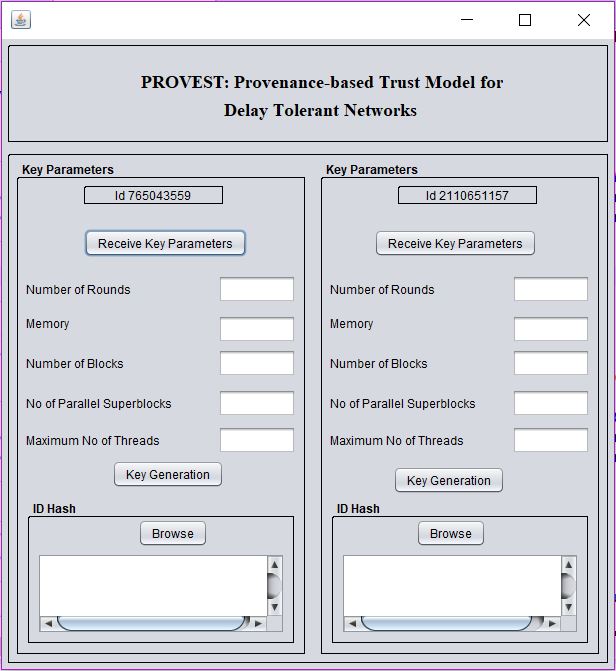


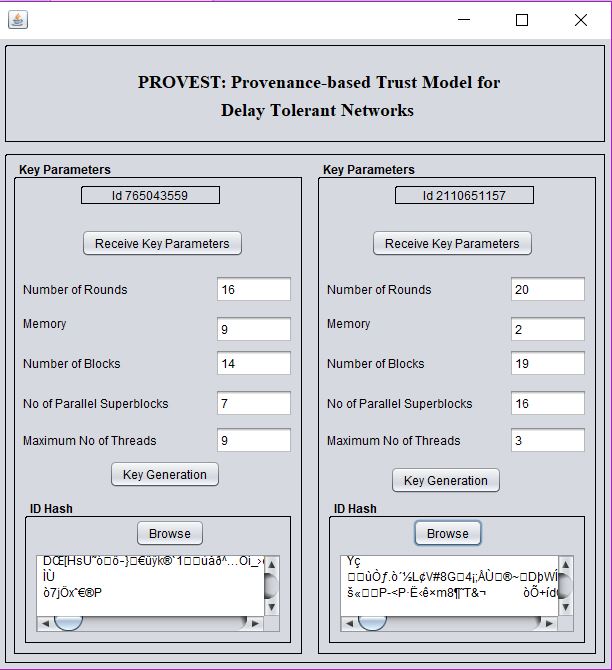


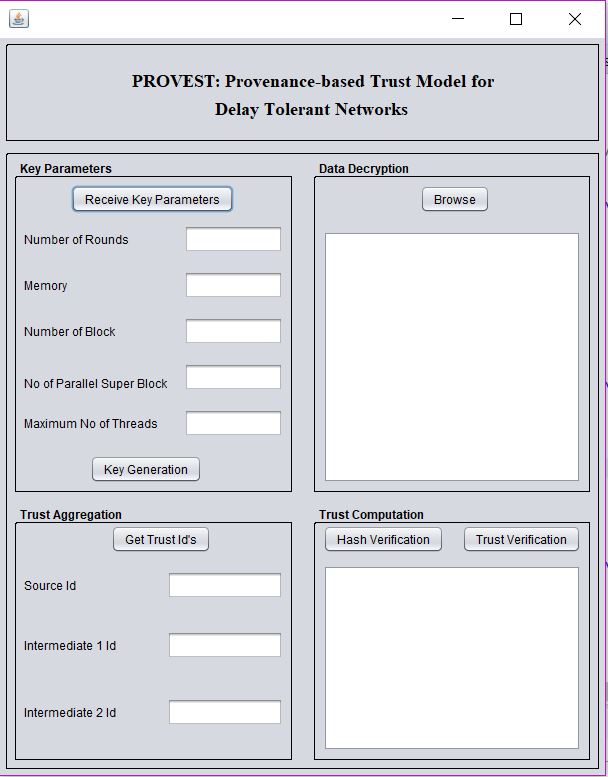
****

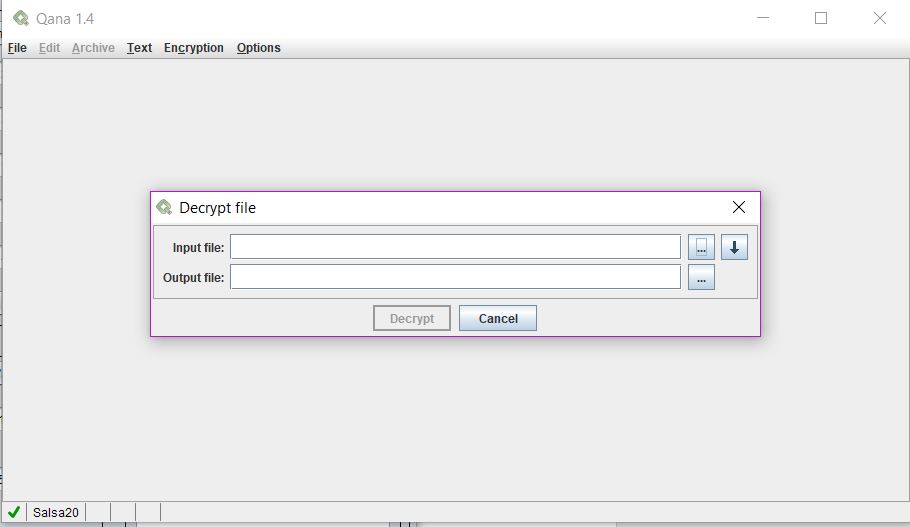
****

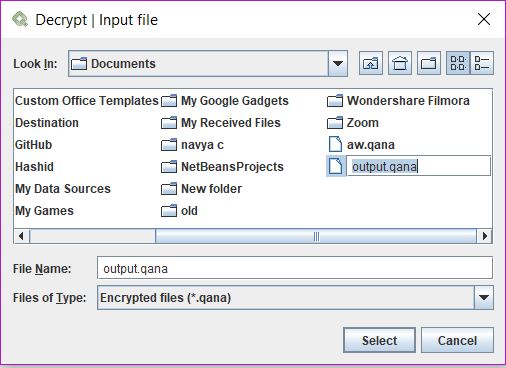


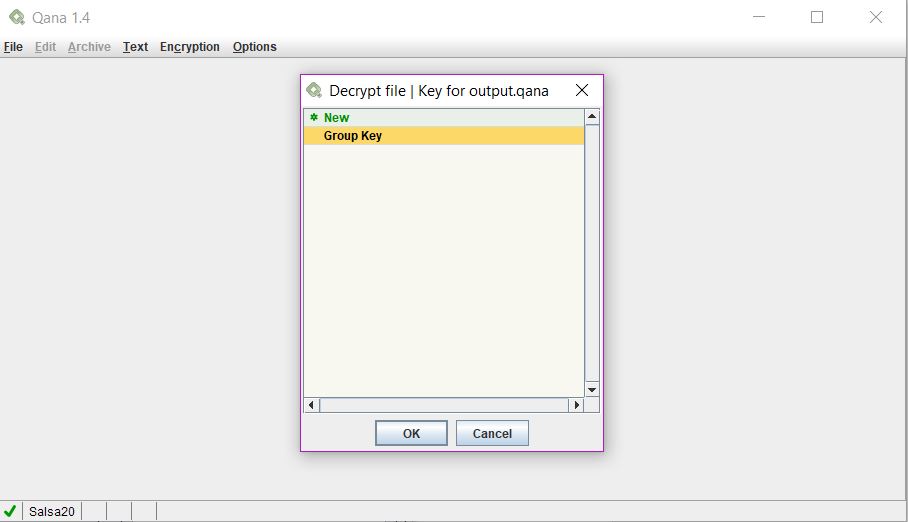


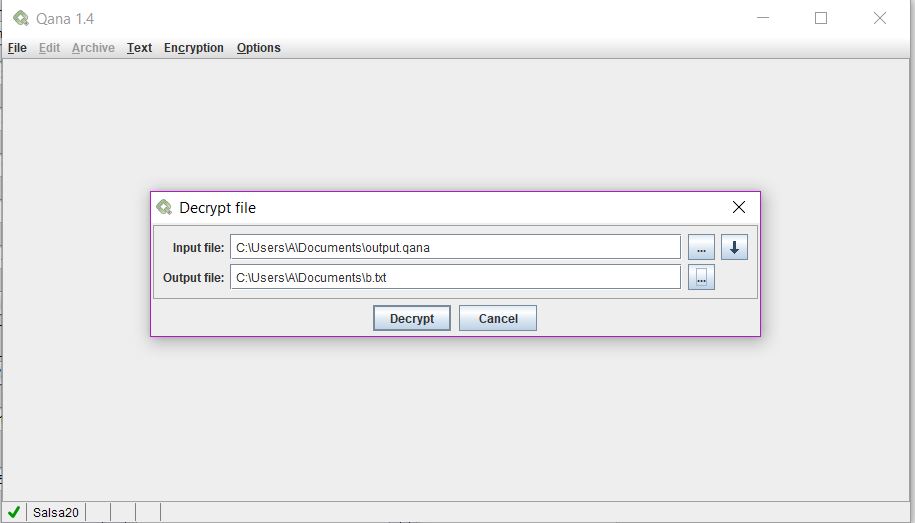


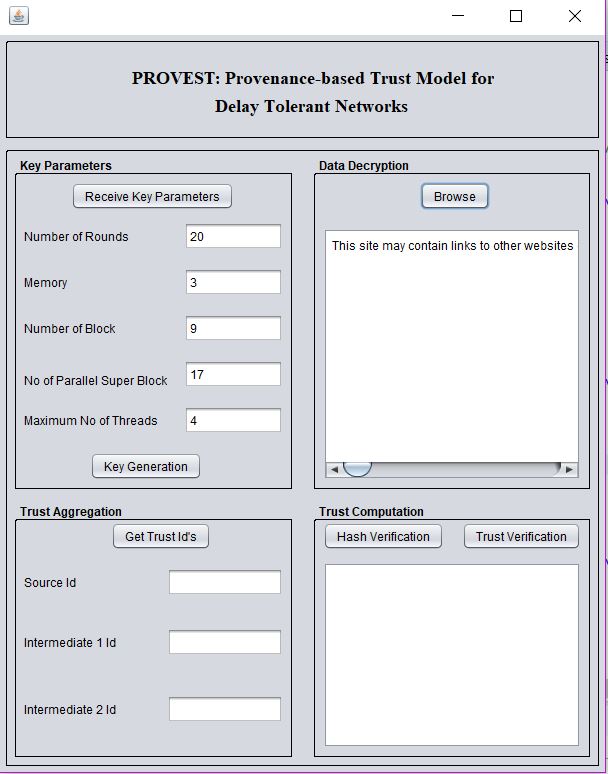


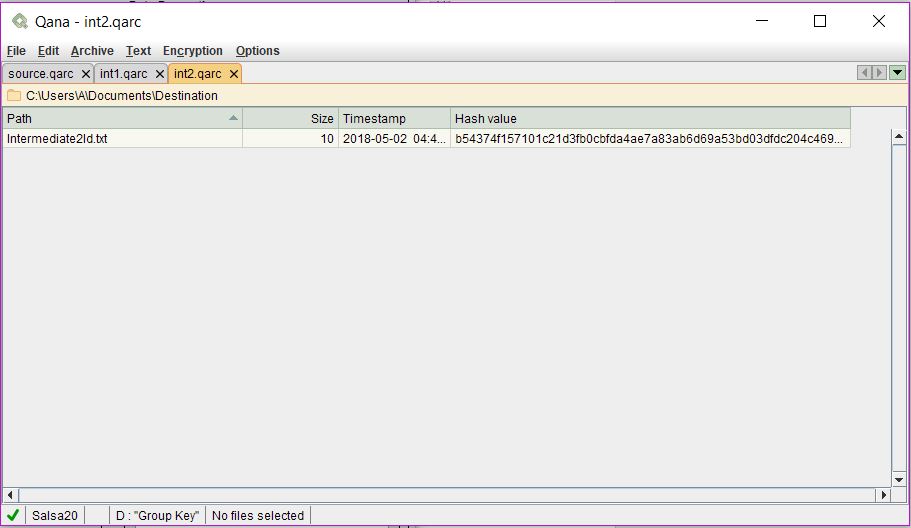
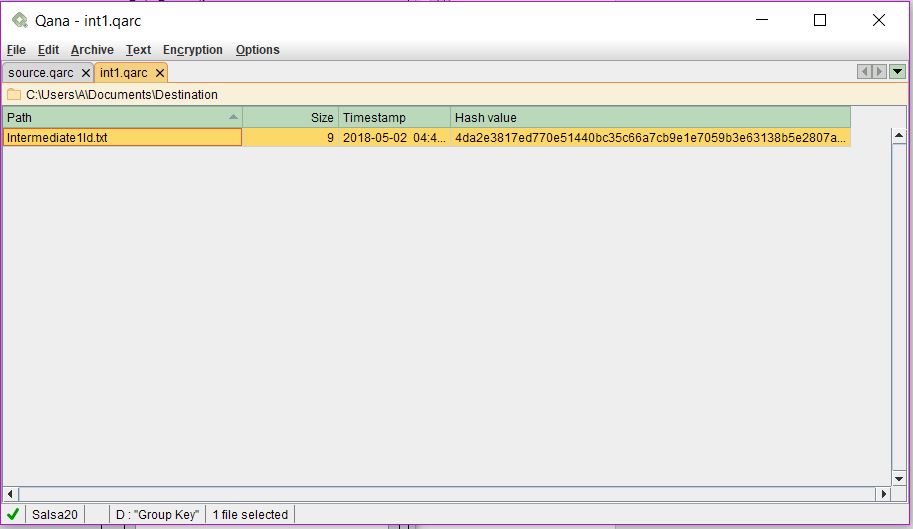
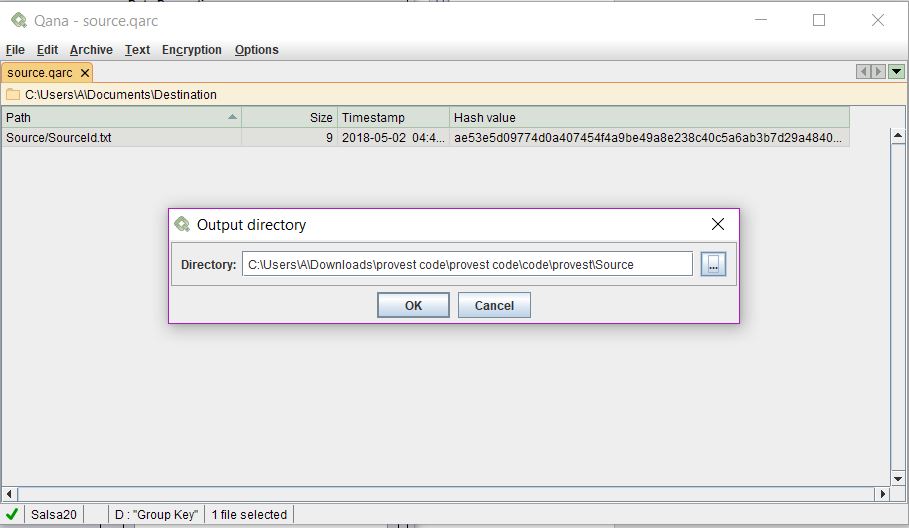
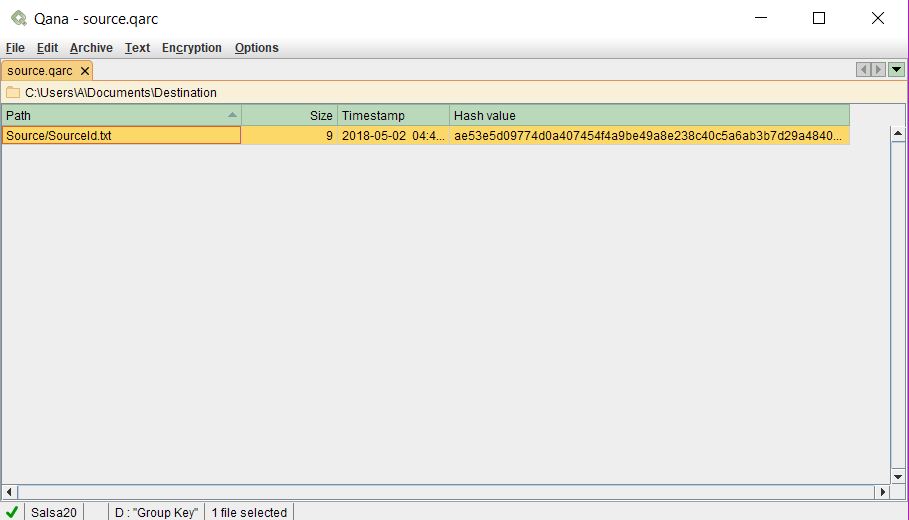


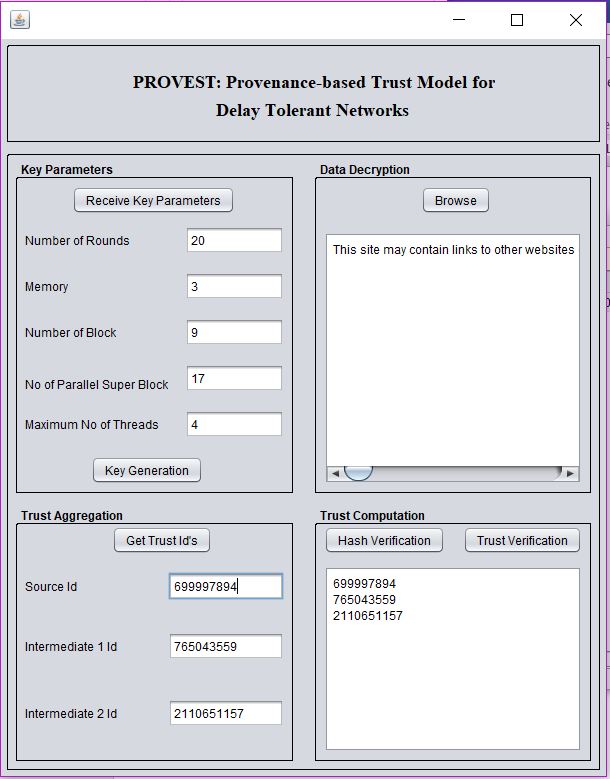


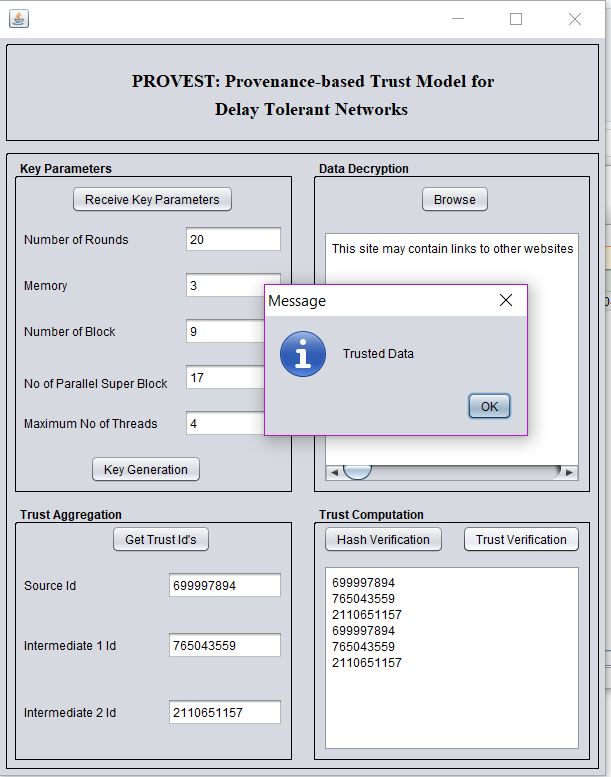












**CHAPTER 7**

**CONCLUSION**

Trust Verification in a DTN is the core aspect of the overall project and it is achieved by considering intermediaries nodes. This project uses source to send the inout file to the destination and then extract it. The intermediaries create their Id’s hash to write to a file which will be read by the destination.

The data is received by the destination which decrypts the data and displays. Finally the archives are extracted using the qana tool and the ids are extracted from hash. This when compared with the verification id’s will say if the data is trusted or Not trusted. We have achieved displaying if the data is trusted or not trusted and thus we can say in a Delay Tolerant Network, our provest model shows the accurate node that can be trusted and interacted with.

If the node is not trusted then we can say it is an external malicious data entering into the network and we can take immediate actions by detecting the node. This saves the intergrity and security of the network.

**REFERENCES**