STAMP: ENABLING PRIVACY-PRESERVING LOCATION PROOFS FOR MOBILE USERS

Authors Name/s A. Vigneysh Aravindh
Department of Computer Science and Engineering
S.R.M University
Chennai 600 026, India
Email ID: - vigneysharavindh@gmail.com

Abstract- Location-based services are quickly becoming immensely popular. In addition to services based on users' current location, many potential services rely on users' location history, or their spatial-temporal provenance. Malicious users may lie about their spatial-temporal provenance without a carefully designed security system for users to prove their past locations. In this paper, we present the Spatial- Temporal provenance Assurance with Mutual Proofs (STAMP) scheme. STAMP is designed for ad-hoc mobile users generating location proofs for each other in a distributed setting. However, it can easily accommodate trusted mobile users and wireless access points. STAMP ensures the integrity and non-transferability of the location proofs and protects users' privacy. A semi-trusted Certification Authority is used to distribute cryptographic keys as well as guard users against collusion by a light-weight entropy-based trust evaluation approach. Our prototype implementation on the Android platform shows that STAMP is low-cost in terms of computational and storage resources. Extensive simulation experiments show that our entropy-based trust model is able to achieve high collusion detection accuracy.

Keywords— Stamp, enabling, privacy-preserving location proofs,

1. INTRODUCTION

1.1 GENERAL

The explosive growth of Internet-capable and location aware mobile devices and the surge in social network usage are fostering collaborative information generation and sharing on an unprecedented scale. In particular, IDC believes that total worldwide

Smartphone shipments will reach 659.8 million units in 2012 and will grow at a CAGR of 18.6 percent until 2016.1 Almost all smartphones have cellular/ Wi-Fi Internet access and can always acquire their precise locations via pre-installed positioning software. Also owing to the growing popularity of social networks, it is more and more convenient and motivating for mobile users to share with others their experience with all kinds of points of interests (POIs) such as bars, restaurants, grocery stores, coffee shops, and hotels. Meanwhile, it becomes commonplace for people to perform various spatial POI queries at online location-based service providers (LBSPs) such as Google and Yelp.

1.2 OBJECTIVE.

This paper focuses on spatial top-k queries, and the term “spatial” will be omitted hereafter for brevity. We observe two essential drawbacks with current top-k query services. First, individual LBSPs often have very small data sets comprising POI reviews. This would largely affect the usefulness and eventually hinder themore prevalent use of spatial top-k query services

1.3 DESCRIPTION

The data sets at individual LBSPs maynot cover all the Italian restaurants within a search radius. Additionally, the same restaurant may receive diverse ratings at different LBSPs, so users may get confused by very

different query results from different LBSPs for the same query. A leading reason for limited data sets at individual LBSPs is that people tend to leave reviews for the same POI at one or at most only a few LBSPs's websites which they often visit. Second, LBSPs may modify their data sets by deleting some reviews or adding fake reviews and return tailored query results in favor of the restaurants that are willing to pay or against those that refuse to pay.2 Even if LBSPs are not malicious, they may return unfaithful query results under the influence of various attacks such as the Sybil attack [2], [3] whereby the same attacker can submit many fake reviews for the same POI. In either case, top-k query users may be misled by the query results to make unwise decisions. A promising solution to the above two issues is to introduce some trusted data collectors as the central hubs for collecting POI reviews. In particular, data collectors can offer various incentives, such as free coffee coupons, for stimulating review submissions and then profit by selling the review data to individual LBSPs. Instead of submitting POI reviews to individual LBSPs, people. Similar misbehaviour has been widely reported for the web-search industry. Data contributors) can now submit them to a few data collectors to earn rewards. The data sets maintained by data collectors can thus be considered the union of the small data sets currently at individual LBSPs. Such centralized data collection also makes it much easier and feasible for data collectors to employ sophisticated defences, such as [2], [3], to filter out fake reviews from malicious entities like Sybil attackers. Data collectors can be either new service providers or more preferably existing ones with a large user base, such as Google, Yahoo, Facebook, Twitter, and MSN. Many of these service providers (e.g., Google) have already been collecting reviews from their users and offered open APIs for exporting selected data from their systems. We postulate that they may act as location-based data collectors and sellers if sound techniques and business models are in place. The above system model is also highly beneficial for LBSPs. In particular, they no longer need struggle to solicit faithful user reviews, which is often a daunting task especially for small/medium-scale LBSPs. Instead, they can focus their limited resources on developing appealing functionalities (such as driving directions and aerial photos) combined with the high-quality review data purchased from data collectors. The query results they can provide will be much more trustworthy, which would in turn help them attract more and more users. This system model thus can greatly help lower the entrance bar for new LBSPs without sufficient funding and thus foster the prosperity of location- based services and applications. A main challenge for realizing the appealing system above is how to deal with untrusted and possibly malicious LBSPs. Specifically, malicious LBSPs may still modify the data sets from data collectors and provide biased top-k query results in favor of POIs willing to pay. Even worse, they may falsely claim generating query results based on the review data from trusted data collectors which they actually did not purchase. Moreover, nonmalicious LBSPs may be compromised to return fake top-k query results.

In this paper, we propose three novel schemes to tackle the above challenge for fostering the practical deployment and wide use of the envisioned system. The key idea of our schemes is that the data collector precomputes and authenticates some auxiliary information (called authenticated hints) about its data set, which will be sold along with its data set to LBSPs. To faithfully answer a top-k query, a LBSP need return the correct top-k POI data records as well as proper authenticity and correctness proofs constructed from authenticated hints. The authenticity proof allows the query user to confirm that the query result only consists of authentic data records from the trusted data collector's data set, and the correctness proof enables the user to verify that the returned top-k POIs are the true ones satisfying the query. The first two schemes both target snapshot top-k queries but differ in how authenticated hints are precomputed and how authenticity and correctness proofs are constructed and verified as well as the related communication and computation overhead. The third scheme, built upon

the first scheme, realizes efficient and verifiable moving top-k queries. The efficacy and efficiency of our schemes are thoroughly analyzed and evaluated through detailed simulation studies. The rest of this paper is organized as follows. Section 2 discusses the related work, and Section 3 gives the problem formulation. Section 4 presents two schemes for secure snapshot top-k query processing, which are extended for secure moving top-k query processing in Section 5. All the schemes are then thoroughly analyzed in Section 6 and evaluated via detailed simulations in Section 7. This paper is finally concluded in Section 8.

2. SYSTEM ANALYSIS

2.1 LITERATURE SURVEY

Title: -"Secure Top-k Query Processing via Untrusted Location-Based Service Providers"

Author: R. Zhang, Y. Zhang, and C. Zhang

Year: 2016

Description:- This paper considers a novel distributed system for collaborative location-based information generation and sharing which become increasingly popular due to the explosive growth of Internet-capable and location-aware mobile devices. The system consists of a data collector, data contributors, location-based service providers (LBSPs), and system users. The data collector gathers reviews about points-of-interest (POIs) from data contributors, while LBSPs purchase POI data sets from the data collector and allow users to perform spatial top-k queries which ask for the POIs in a certain region and with the highest k ratings for an interested POI attribute. In practice, LBSPs are untrusted and may return fake query results for various bad motives, e.g., in favor of POIs willing to pay. This paper presents three novel schemes for users to detect fake spatial snapshot and moving top-k query results as an effort to foster the practical deployment and use of the proposed system. The efficacy and efficiency of our schemes are thoroughly analyzed and evaluated

Title: - SybilGuard: Defending against Sybil Attacks via Social Networks

Author: H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman

Year: 2014

Description: Peer-to-peer and other decentralized, distributed systems are known to be particularly vulnerable to sybil attacks. In a Sybil attack, a malicious user obtains multiple fake identities and pretends to be multiple, distinct nodes in the system. By controlling a large fraction of the nodes in the system, the malicious user is able to "out vote" the honest users in collaborative tasks such as Byzantine failure defences. This paper presents SybilGuard, a novel protocol for limiting the corruptive influences of sybil attacks. Our protocol is based on the "social network" among user identities, where an edge between two identities indicates a human-established trust relationship. Malicious users can create many identities but few trust relationships. Thus, there is a disproportionately-small "cut" in the graph between the sybil nodes and the honest nodes. SybilGuard exploits this property to bound the number of identities a malicious user can create. We show the effectiveness of SybilGuard both analytically and experimentally.

Title: - SybilLimit: A Near-Optimal Social Network Defence against Sybil Attacks

Author: H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao

Year: 2015

Description:- Decentralized distributed systems such as peer-to-peer systems are particularly vulnerable to sybil attacks, where a malicious user pretends to have multiple identities (called sybil nodes). Without a trusted central authority, defending against sybil attacks is quite challenging. Among the small number of decentralized approaches, our recent SybilGuard protocol [H. Yu et al., 2006] leverages a key insight on social networks to bound the number of sybil nodes accepted. Although its direction is promising, SybilGuard can allow a large number of sybil nodes to be accepted. Furthermore, SybilGuard assumes that

social networks are fast mixing, which has never been confirmed in the real world. This paper presents the novel SybilLimit protocol that leverages the same insight as SybilGuard but offers dramatically improved and near-optimal guarantees. The number of sybil nodes accepted is reduced by a factor of ominus(radicn), or around 200 times in our experiments for a million-node system. We further prove that SybilLimit's guarantee is at most a log n factor away from optimal, when considering approaches based on fast-mixing social networks. Finally, based on three large-scale real-world social networks, we provide the first evidence that real-world social networks are indeed fast mixing. This validates the fundamental assumption behind SybilLimit's and SybilGuard's approach.

Title:- Providing Database as a Service

Author:- H. Hacig€um€us, S. Mehrotra, and B. Iyer

Description:- We explore a novel paradigm for data management in which a third party service provider hosts "database as a service", providing its customers with seamless mechanisms to create, store, and access their databases at the host site. Such a model alleviates the need for organizations to purchase expensive hardware and software, deal with software upgrades, and hire professionals for administrative and maintenance tasks which are taken over by the service provider. We have developed and deployed a database service on the Internet, called NetDB2, which is in constant use. In a sense, a data management model supported by NetDB2 provides an effective mechanism for organizations to purchase data management as a service, thereby freeing them to concentrate on their core businesses. Among the primary challenges introduced by "database as a service" are the additional overhead of remote access to data, an infrastructure to guarantee data privacy, and user interface design for such a service. These issues are investigated. We identify data privacy as a particularly vital problem and propose alternative solutions based on data encryption. The paper is meant as a challenge for the database community to explore a rich set of research issues that arise in developing such a service.

2.2 EXISTING SYSTEM

Existing schemes which require multiple trusted or semi-trusted third parties, STAMP requires only Single semi-trusted third party which can be embedded in a Certificate Authority (CA). We design our system with an objective of protecting users' anonymity and location privacy. No parties other than verifiers could see both a user's identity and STP information (verifiers need both identity and STP information in order to perform verification and provide services). Users are given the flexibility to choose the location granularity level that is revealed to the verifier. We examine two type s of collusion attacks: (1) A user who is at an intended location masquerades s another colluding user and obtains STP proofs for . This attack has never been addressed in any existing STP proof schemes. (2) Colluding users mutually generate fake STP proofs for each other. There have been efforts to address this type of collusion. However, existing solutions suffer from high computational cost and low scalability. Particularly, the latter collusion scenario is in fact the challenging Terrorist Fraud attack, which is a critical issue for our targeted system, but none of the existing systems has addressed it. We integrate the Bussard-Bagga distance bounding protocol into STAMP to protect our scheme against this collusion attack. Collusion scenario (1) is hard to prevent without a trusted third party. To make our system resilient to this attack, we propose an entropy-based trust model to detect the collusion scenario. We implemented STAMP on the Android platform and carried out extensive validation experiments. The experimental results show that STAMP requires low computational overhead.

DISADVANTAGES:

Solutions suffer from high computational cost and low scalability. Particularly, the latter collusion scenario is in fact the challenging Terrorist Fraud attack, which is a critical issue for our targeted system, but none of the existing systems has addressed it.

2.3 PROPOSED SYSTEM

In this paper, we propose an STP proof scheme named Spatial-Temporal provenance Assurance with Mutual Proofs (STAMP). STAMP aims at ensuring the integrity and non-transferability of the STP proofs, with the capability of protecting users' privacy. Most of the existing STP proof schemes rely on wireless infrastructure (e.g., WiFi APs) to create proofs for mobile users. However, it may not be feasible for all types of applications, e.g., STP proofs for the green commuting and battlefield examples certainly cannot be obtained from wireless APs. To target a wider range of applications, STAMP is based on a distributed architecture. Co-located mobile devices mutually generate and endorse STP proofs for each other, while at the same time it does not eliminate the possibility of utilizing wireless infrastructures as more trusted proof generation sources. In addition, in contrast to most of the existing schemes which require multiple trusted or semi-trusted third parties, STAMP requires only a single semi-trusted third party which can be embedded in a Certificate Authority (CA). We design our system with an objective of protecting users' anonymity and location privacy. No parties other than verifiers could see both a user's identity and STP information (verifiers need both identity and STP information in order to perform verification and provide services). Users are given the flexibility to choose the location granularity level that is revealed to the verifier.

ADVANTAGES:

- Our security analysis shows that STAMP achieves the security and privacy objectives.
- Our implementation on Android smartphones indicates that low computational and storage resources are required to execute STAMP.
- Extensive simulation results show that our trust model is able to attain a high balanced accuracy with appropriate choices of system parameters.

3. PROJECT DESCRIPTION

3.1 GENERAL

STAMP aims at ensuring the integrity and non-trans-fesability of the STP proofs, with the capability of protecting user's privacy.

3.2 PROBLEM DEFINITION

The system that is most closely related to our work is a location proof system that is also based on co-located mobile devices mutually generating location proofs. In order to protect privacy, the knowledge of private information is separately distributed to three parties: a location proof server, a CA, and the verifier. Periodically changed pseudonyms are used by the mobile devices to protect their real identities from each other, and from the location proof server. We believe the location proof server is redundant for accomplishing the goals. Periodically changed pseudonyms incurs high operational overhead because of the requirement for highly cautious management and scheduling. Dummy proofs have to be regularly generated in order to achieve the privacy properties, which also incurs high communication and storage overhead.

3.3 METHODOLOGIES

Enabling Privacy-Preserving Location Proofs for Mobile Users Use the DES Algorithm. The user searched location are encrypted and that encrypted can be viewed only by the verifier and certificate authority, even the verifier and the certificate authority can view only in the form of encryption. Thus the location cannot be found by the third party or unknown person.

3.3.1 MODULE DESCRIPTION & MODULE DIAGRAMS

MODULES

- **User Module**

- **Sign In Module**
- **Sign Up Module**

- **Verifier Module**
- **Certificate Authority Module**

4. SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 512 Mb.

4.2 SOFTWARE REQUIREMENTS:

- Operating system : Windows XP/7.
- Coding Language : JAVA/J2EE
- IDE: Netbeans 7.4
- Database : MYSQL

5. SYSTEM DESIGN

5.1 GENERAL

System design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.

For assessing user requirements, an SRS (Software Requirement Specification) document is created whereas for coding and implementation, there is a need of more specific and detailed requirements in software terms. The output of this process can directly be used into implementation in programming languages.

Software design is the first step in SDLC (Software Design Life Cycle), which moves the concentration from problem domain to solution domain. It tries to specify how to fulfill the requirements mentioned in SRS.

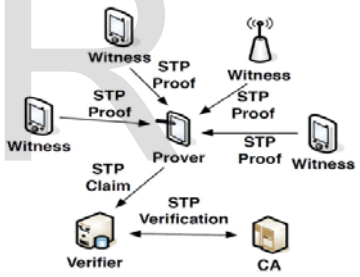


Fig 1.0 SYSTEM DESIGN

5.2.1 SYSTEM ARCHITECTURE

Fig 1.1 DEPLOYMENT DIAGRAM

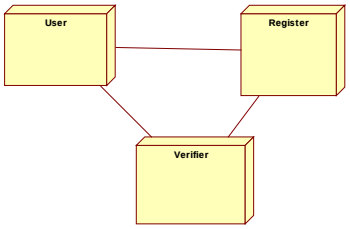
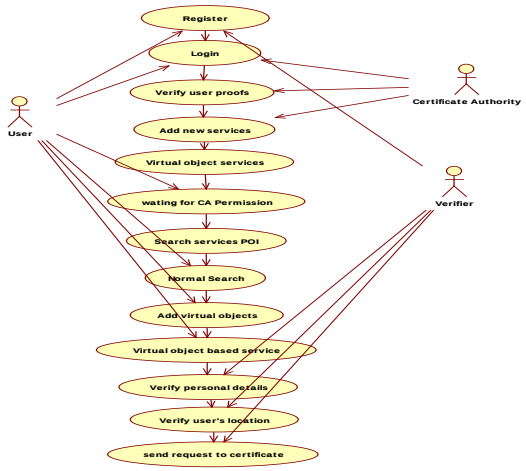


Fig1.2 USE CASE DIAGRAM



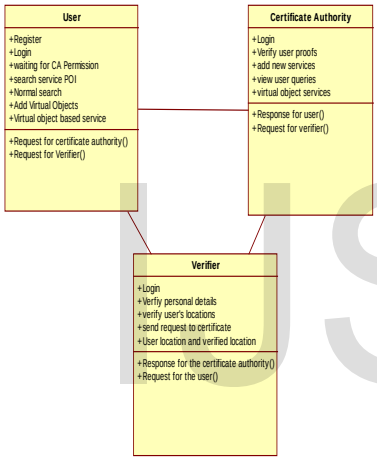


Fig1.3 Class Diagram

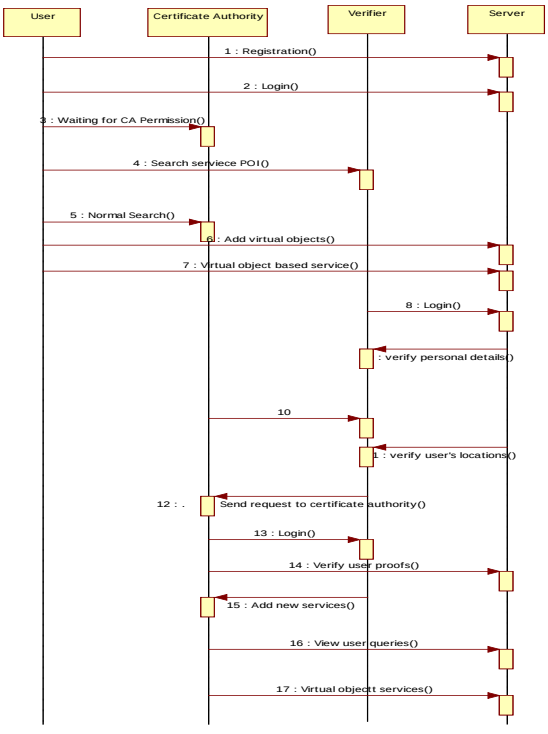


Fig1.4 Sequence diagram

6. SOFTWARE TESTING

6.1 GENERAL

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 DEVELOPING METHODOLOGIES

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.

The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

6.3 TYPES OF TESTS

6.3.1 Unit Test

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform

basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.3.2 Functional Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted

Invalid Input : identified classes of invalid input must be rejected

Functions : identified functions must be exercised

Output : identified classes of application outputs must be exercised

Systems/Procedures : interfacing systems or procedures must be invoked.

6.3.3 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.3.4 Performance Test

The performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

6.3.5 Integration Test

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

6.3.6 Acceptance Test

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Acceptance Testing for Data Synchronization:

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node.
- The Route add operation is done only when there is a Route request in need
- The status of Nodes information is done automatically in the Cache Updating process.

6.3.7 Build the test plan

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identify the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

7. APPLICATION

7.1 GENERAL

This session gives the details of our application usage in secure Image transmission and its Usefulness.

7.2. APPLICATION

USE IN MODERN PRINTERS

The larger the Target Image (in binary data, the number of bits) relative to the hidden Encrypted Image, the easier it is to hide the latter. For this reason, digital pictures (which contain large amounts of data) are used to hide messages on the Internet and on other communication media. It is not clear how common this actually is. For example: a 24-bit bitmap uses 8 bits to represent each of the three colour values (red, green, and blue) at each pixel. The blue alone has 28 different levels of blue intensity. The difference between 11111111 and 11111110 in the value for blue intensity

is likely to be undetectable by the human eye. Therefore, the least significant bit can be used more or less undetectably for something else other than colour information. If this is repeated for the green and the red elements of each pixel as well, it is possible to encode one byte of Encrypted Image for every three pixels. Some modern computer printers use steganography, including HP and Xerox brand colour laser printers. These printers could print the Dots which are Actually the Encrypted Byte Array which can be recovered on scanning the Image.

USE IN MOBILE APPLICATIONS

Can be used as an Android / iOS Application to to secure images so that if Users do not want third party apps or users to access their content, they could make use of the App to Encrypt and Transform the Image into something that is irrelevant to the context of the Original Image. That way, the third party Apps or users will not be able to guess that the image has been transformed,hence they wouldn't be interested in that Image at all. Lots of Online Blackmailing and abuse can be prevented with the safety features this application provides. It's a viable method to ensure Privacy and Protection of personal Content.

CONCLUSION

In this paper we have presented STAMP, which aims at providing security and privacy assurance to mobile users' proofs for their past location visits. STAMP relies on mobile devices in vicinity to mutually generate location proofs or uses wireless APs to generate location proofs. Integrity and non-transferability of location proofs and location privacy of users are the main design goals of STAMP.We have specifically dealt with two collusion scenarios: P-P collusion and P-W collusion. To protect against P-P collusions, we integrated the Bussard-Bagga distance bounding protocol into the design of STAMP. To detect P-W collusion, we proposed an entropy-based trust model to evaluate the trust level of claims of the past location visits. Our security analysis shows that STAMP achieves the security and privacy objectives. Our implementation on Android smartphones indicates that low computational and storage resources are required to execute STAMP. Extensive simulation results show that our trust model is able to attain a high balanced accuracy (> 0.9) with appropriate choices of system parameters.

APPENDIX 1

SAMPLE CODING

```
public class EncryptDecrypt {

   public static void main(String arg[]) {
      EncryptDecrypt edObj = new EncryptDecrypt();
      //String str =
"16316916916315415716315416621920722321920722321922723 1";
      String str =
"abcdefghijklmnopqrstuvwwwwxyyzzzz";
      String enCryptStr = edObj.Encrypt(str);
      System.out.println(" Encrypted string : " +
enCryptStr);
      System.out.println(" Decrypted string : " +
edObj.Decrypt(enCryptStr));
   }

   //Encryption method
   public String Encrypt(String passwd) {
      int i, len, asci_code, temp_val = 0;
      char ch;
      String out_str = new String();
      len = passwd.length();
      for (i = 0; i < len; i++) {
         ch = passwd.charAt(i);
         asci_code = getAscii(ch);
         if (i < len / 2) {
            temp_val = (asci_code * 3) + 10;
         } else {
            temp_val = (asci_code * 4) + 11;
         }
         out_str = out_str + temp_val;
      }
      return out_str;
   }

   public char getCharec(int i) {
      /*for(int x=0;x<=arr.length;x++) {
      if(arr[x] == i) {
      return (chars.charAt(x));
      }
```

```
        }
        return 0;*/
        return (char) i;
    }

    public int getAscii(char c) {
        /*for(int x=0;x<=chars.length();x++){
        if(chars.charAt(x) == c) {
        return (arr[x]);
        }
        }
        return 0;*/
        return (int) c;
    }

    /**
     * this function is to decrypt the password
     * @param string password
     */
    public String Decrypt(String passwd) {
        int i, len, asci_code, temp_val = 0;
        String out_str = new String();
        String char_str = new String();
        len = passwd.length();
        for (i = 0; i < len; i += 3) {
            if ((i + 3) <= len) {
                char_str = passwd.substring(i, i + 3);
                asci_code = Integer.parseInt(char_str);
                if (i < (len - 2) / 2) {
                    temp_val = ((asci_code - 10) / 3);
                } else {
                    temp_val = ((asci_code - 11) / 4);
                }
            }
            out_str = out_str + this.getCharec(temp_val);
        }
        return (out_str);
    }
}
```

```
/*
 * To change this license header, choose License
Headers in Project Properties.
 * To change this template file, choose Tools |
Templates
 * and open the template in the editor.
 */

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author Sai Krishna
 */
public class insert extends HttpServlet {

    /**
     * Processes requests for both HTTP
<code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific
error occurs
     * @throws IOException if an I/O error occurs
     */
    String name,add,lati,longi,map,cate;
    protected void processRequest(HttpServletRequest
request, HttpServletResponse response)
            throws ServletException, IOException
    {
        response.setContentType("text/html;charset=UTF-
8");
        PrintWriter out = response.getWriter();
        {
            try {
                name=request.getParameter("name");
                add=request.getParameter("add");
                lati=request.getParameter("lati");
                longi=request.getParameter("longi");
                map=request.getParameter("map");
                cate=request.getParameter("cate");
                Class.forName("com.mysql.jdbc.Driver");
                Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:
3306/loc","root","root");
                String sql="insert into det
(name,address,lav,lov,map,cate) values
('"+name+"','"+add+"','"+lati+"','"+longi+"','"+map+"','"
+cate+"')";
                PreparedStatement
pss=con.prepareStatement(sql);
                int executeUpdate = pss.executeUpdate();
                out.println("<script>"
                                                +"alert('Details
are inserted successfully')"
                                                +"</script>");
                RequestDispatcher
rd=request.getRequestDispatcher("/insert.jsp");
                        rd.include(request, response);

            } catch (Exception ex)
            {

Logger.getLogger(Reg.class.getName()).log(Level.SE
VERE, null, ex);
                out.println("<script>"
                                                +"alert('Please
Check your Database Connection')"
                                                +"</script>");
                RequestDispatcher
rd=request.getRequestDispatcher("/insert.jsp");
                        rd.include(request, response);
            }
        }
```

```
    }

    // <editor-fold defaultstate="collapsed"
desc="HttpServlet methods. Click on the + sign on the
left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific
error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific
error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

```
/*
 * To change this license header, choose License
Headers in Project Properties.
 * To change this template file, choose Tools |
Templates
 * and open the template in the editor.
 */

import java.io.IOException;
import java.io.PrintWriter;
import java.security.SecureRandom;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 *
 * @author Sai Krishna
 */
public class login extends HttpServlet {

    /**
     * Processes requests for both HTTP
<code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific
error occurs
     * @throws IOException if an I/O error occurs
     */

    String name,pass1,p="";
    protected void processRequest(HttpServletRequest
request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-
8");
        PrintWriter out = response.getWriter();

            try {
                name=request.getParameter("user");
                pass1=request.getParameter("pass");

                Class.forName("com.mysql.jdbc.Driver");
                Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:
3306/loc", "root","root");
                String q="select * from reg where
name='"+name+"' and pass='"+pass1+"'";
                PreparedStatement
ps=con.prepareStatement(q);
                ResultSet rs = ps.executeQuery();
                if(rs.next())
                {
                  out.println("<script>"
                          +"alert('Valid User')"
                          +"</script>");
                   HttpSession session=request.getSession();
                   session.setAttribute("name", name);
                   session.setAttribute("pass", pass1);
                   RequestDispatcher
rd=request.getRequestDispatcher("/userhome.jsp");
                    rd.include(request, response);
                }
                else
if(name.equals("lbs")&&pass1.equals("lbs"))
```

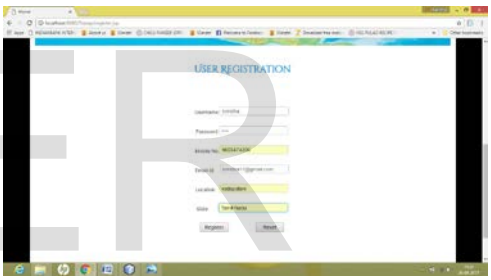
```
            {
                out.println("<script>"
                        +"alert('Welcome Admin')"
                        +"</script>");
                RequestDispatcher
rd=request.getRequestDispatcher("/adminhome.jsp");
                RequestDispatcher
rd=request.getRequestDispatcher("/index.jsp");
                rd.include(request, response);
                }
        } catch (Exception ex)

            {

Logger.getLogger(login.class.getName()).log(Level.SE
VERE, null, ex);
            }
    }

    // <editor-fold defaultstate="collapsed"
desc="HttpServlet methods. Click on the + sign on the
left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific
error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific
error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
```

```
            rd.include(request, response);
        }
        else{
            out.println("<script>"
                    +"alert('Invalid Login')"
                    +"</script>");
}
```

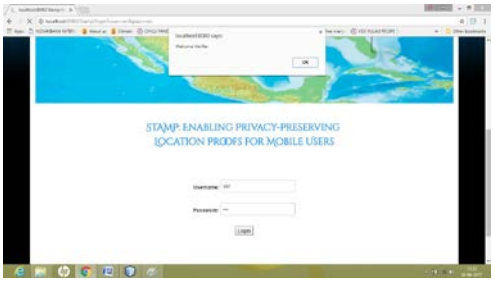
APPENDIX 2

SCREENSHOTS

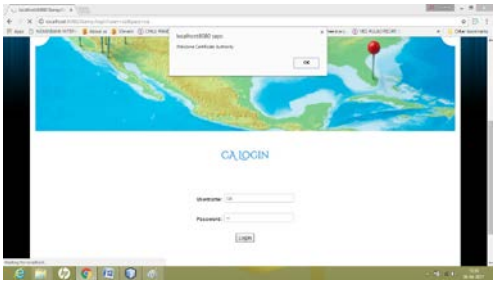
A2.1 User Registration



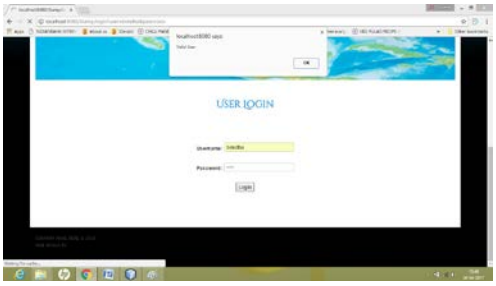
A2.2 Verifier Login



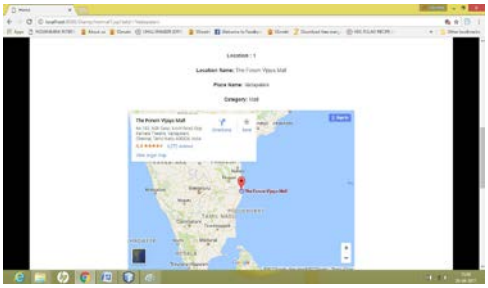
A2.3 Certificate Authority Login

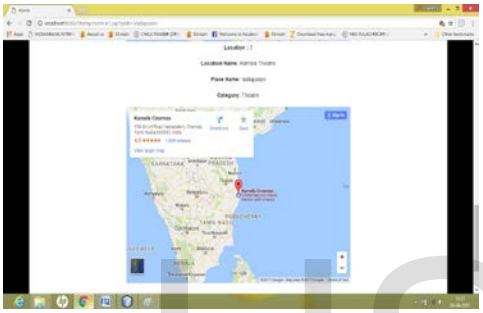


A2.4 User Login



A2.5 Location Enabled





REFERENCES

[1] S. Saroiu and A. Wolman, "Enabling new mobile applications with location proofs," in Proc. ACM HotMobile, 2009, Art. no. 3.

[2] W. Luo and U. Hengartner, "VeriPlace: A privacy-aware location proof architecture," in Proc. ACM GIS, 2010, pp. 23–32.

[3] Z. Zhu and G. Cao, "Towards privacy-preserving and colluding-resistance in location proof updating system," IEEE Trans. Mobile Comput., vol. 12, no. 1, pp. 51–64, Jan. 2011.

[4] N. Sastry, U. Shankar, and D. Wagner, "Secure verification of location claims," in Proc. ACM WiSe, 2003, pp. 1–10.

[5] R. Hasan and R. Burns, "Where have you been? secure location provenance for mobile devices," CoRR 2011.

[6] B. Davis, H. Chen, and M. Franklin, "Privacy preserving alibi systems," in Proc. ACM ASIACCS, 2012, pp. 34–35.

[7] I. Krontiris, F. Freiling, and T. Dimitriou, "Location privacy in urban sensing networks: Research challenges and directions," IEEE Wireless Commun., vol. 17, no. 5, pp. 30–35, Oct. 2010.

[8] Y. Desmedt, "Major security problems with the 'unforgeable' (feige)- fiat-shamir proofs of identity and how to overcome them," in Proc. SecuriCom, 1988, pp. 15–17.

[9] L. Bussard and W. Bagga, "Distance-bounding proof of knowledge to avoid real-time attacks," in Security and Privacy in the Age of Ubiquitous Computing. New York, NY, USA: Springer, 2005.

[10] B. Waters and E. Felten, "Secure, private proofs of location," Department of Computer Science, Princeton University, Princeton, NJ, USA, Tech. Rep., 2003.

[11] X. Wang et al., "STAMP: Ad hoc spatial-temporal provenance assurance for mobile users," in Proc. IEEE ICNP, 2013, pp. 1–10.

[12] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity-a proposal for terminology," in Designing Privacy Enhancing Technologies. New York, NY, USA: Springer, 2001.

[13] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole attacks in wireless networks," IEEE J. Sel. Areas Commun., vol. 24, no. 2, pp. 370–380, Feb. 2006.

[14] S. Halevi and S. Micali, "Practical and provably-secure commitment schemes from collision-free hashing," in Proc. CRYPTO, 1996, pp. 201–215.

[15] I. Damgård, "Commitment schemes and zero-knowledge protocols," in Proc. Lectures Data Security, 1999, pp. 63–86.

[16] I. Haitner and O. Reingold, “Statistically-hiding commitment from any one-way function,” in Proc. ACM Symp. Theory Comput., 2007, pp. 1–10.

[17] D. Singelee and B. Preneel, “Location verification using secure distance bounding protocols,” in Proc. IEEE MASS, 2005.