Volume 3, Issue 7, July 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Anonymous Authentication Backward Unlink ability, Subjective Blacklisting, rate-limited Anonymous Connections (secure system)

B.Srinivasulu[#1],S.Sukanya[#2],A.Bhima Sanakram#[3],R.V Sudhakar#[4]

JNTU , Hyderabad , India

Abstract -We present a secure system called Nymble, which provides all the following properties: anonymous authentication, backward unlinkability, subjective blacklisting, fast authentication speeds, rate-limited anonymous connections, revocation auditability (where users can verify whether they have been blacklisted), and also addresses the Sybil attack to make its deployment practical. In Nymble, users acquire an ordered collection of nymbles,a special type of pseudonym, to connect toWebsites. Without additional information, these nymbles are computationally hard to link,4 and hence, using the stream of nymbles simulates anonymous access to services. Web sites, however, can blacklist users by obtaining a seed for a particular nymble, allowing them to link future nymbles from the same user—those used before the complaint remains unlinkable. Servers can therefore blacklist anonymous users without knowledge of their IP addresses while allowing behaving users to connect anonymously. Our system ensures that users are aware of their blacklist status before they present a nymble, and disconnect immediately if they are blacklisted. Although our work applies to anonymizing networks in general, we consider Tor for purposes of exposition. In fact, any number of anonymizing networks can rely on the same Nymble system, blacklisting anonymous users regardless of their anonymizing network(s) of choice.

Keywords: anonymous authentication, backward unlinkability, revocation auditability , Nymble, Dynamism, Sybil attacks.

I. Introduction

Anonymizing networks such as Tor allow users to access Internet services privately by using a series of routers to hide the client's IP address from the server. The success of such networks, however, has been limited by users employing this anonymity for abusive purposes such as defacing popular websites. Website administrators routinely rely on IP-address blocking for disabling access to misbehaving users, but blocking IP addresses is not practical if the abuser routes through an anonymizing network. As a result, administrators block all known exit nodes of anonymizing networks, denying anonymous access to misbehaving and behaving users alike. To address this problem, we present Nymble, a system in which servers can "blacklist" misbehaving users, thereby blocking users without compromising their anonymity. Our system is thus agnostic to different servers' definitions of misbehavior — servers can blacklist users for whatever reason, and the privacy of blacklisted users is maintained.Existing users' credentials must be updated, making it impractical. and if there are any users who are good at programming and hacking can change there IP address and can perform illegal operations in the servers such as google, yahoo etc. where they can change the information present in the servers database which is illegal. And we can't trace these hackers in the existing system. We present a secure system called Nymble, which provides all the following properties: anonymous authentication, backward unlinkability, subjective blacklisting, fast authentication speeds, rate-limited anonymous connections, revocation auditability (where users can verify whether they have been blacklisted), and also addresses the Sybil attack to make its deployment practical .In Nymble, users acquire an ordered collection of nymbles, a special type of pseudonym, to connect to websites. Without additional information, these nymbles are computationally hard to link, and hence using the stream of nymbles simulates anonymous access to services.Websites, however, can blacklist users by obtaining a seed for a particular nymble, allowing them to link future nymbles from the same user — those used before the complaint remain unlinkable. Servers can therefore blacklist anonymous users without knowledge of their IP addresses while allowing behaving users to connect anonymously. Our system ensures that users are aware of their blacklist status before they present a nymble, and disconnect immediately if they are blacklisted. Although our work applies to anonymizing networks in general, we consider Tor for purposes of exposition. In fact, any number of anonymizing networks can rely on the same Nymble system, blacklisting anonymous users regardless of their anonymizing network(s) of choice.

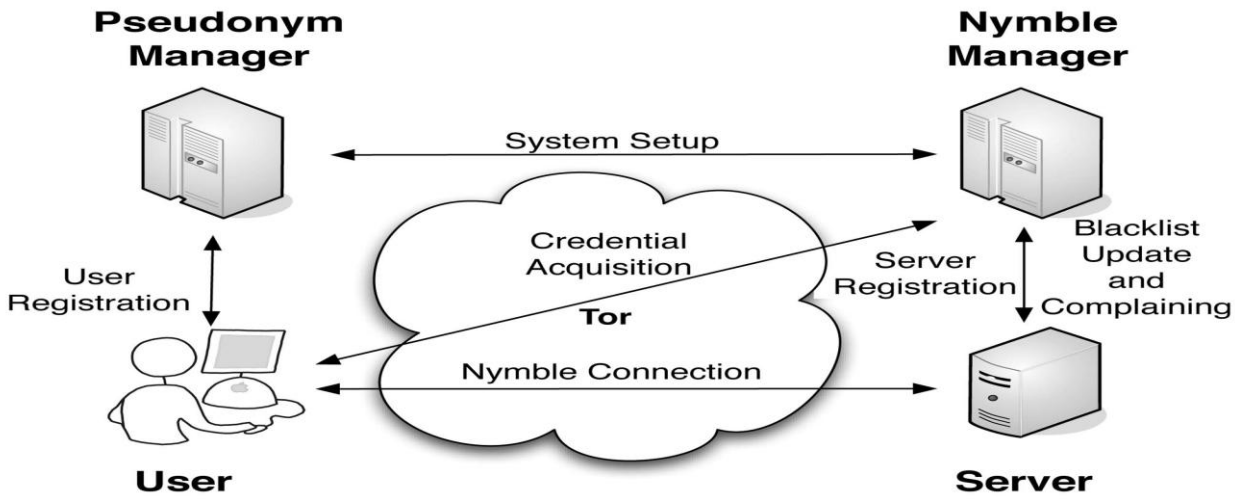
II. SYSTEM OVERVIEW

Anonymizing networks such as Tor route traffic through independent nodes in separate administrative domains to hide a client's IP address. Unfortunately, some users have misused such networks—under the cover of anonymity, users have repeatedly defaced popular Web sites such as Wikipedia. Since Web site administrators cannot blacklist individual malicious

users' IP addresses, they blacklist the entire anonymizing network. Such measures eliminate malicious activity through anonymizing networks at the cost of denying anonymous access to behaving users. In other words, a few "bad apples" can spoil the fun for all . (This has happened repeatedly with Tor.)There are several solutions to this problem, each providing some degree of accountability. In pseudonymous credential systems users log into Websites using pseudonyms, which can be added to a blacklist if a user misbehaves. Unfortunately, thisapproach results in pseudonymity for all users, and weakens the anonymity provided by the anonymizing network. Anonymous credential systems employ group signatures. Basic group signatures allow servers to revoke a misbehaving user's anonymity by complaining to a group manager. Servers must query the group manager for every authentication, and thus, lacks scalability. Traceable signatures allow the group manager to release a trapdoor that allows all signatures generated by a particular user to be traced; such an approach does not provide the backward unlinkability that we desire, where a user's accesses before the complaint remain anonymous

Our Solution:

We present a secure system called Nymble, which provides all the following properties: anonymous authentication,backward unlinkability, subjective blacklisting, fast authentication speeds, rate-limited anonymous connections, revocation auditability (where users can verify whether they have been blacklisted), and also addresses the Sybil attack to make its deployment practical. In Nymble, users acquire an ordered collection of nymbles,a special type of pseudonym, to connect toWebsites. Without additional information, these nymbles are computationally hard to link,4 and hence, using the stream of nymbles simulates anonymous access to services. Web sites, however, can blacklist users by obtaining a seed for a particular nymble, allowing them to link future nymbles from the same user—those used before the complaint remain unlinkable. Servers can therefore blacklist anonymous users without knowledge of their IP addresses while allowing behaving users to connect anonymously. Our system ensures that users are aware of their blacklist status before they present a nymble, and disconnect immediately if they are blacklisted. Although our work applies to anonymizing networks in general, we consider Tor for purposes of exposition. In fact, any number of anonymizing networks can rely on the same Nymble system, blacklisting anonymous users regardless of their anonymizing network(s) of choice.



III. The Working Principle

Initially the language was called as "oak" but it was renamed as "Java" in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

Java is a programmer's language.

Java is cohesive and consistent.

Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.

Finally, Java is to Internet programming where C was to system programming.

Features of Java Security

Every time you that you download a "normal" program, you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scan them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both these concerns by providing a "firewall" between a network application and your computer.

The Byte code

The key that allows the Java to solve the security and portability problems is that the output of Java compiler is Byte code. Byte code is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreter for byte code.

Java Virtual Machine (JVM)

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system.

Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that's has been generated by the compiler will not corrupt the machine that it's loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So byte code verification is integral to the compiling and executing of Java code

Java programming uses to produce byte codes and executes them. The first box indicates that the Java source code is located in a. Java file that is processed with a Java compiler called javac. The Java compiler produces a file called a. class file, which contains the byte code. The Class file is then loaded across the network or loaded locally on your machine into the execution environment is the Java virtual machine, which interprets and executes the byte code.

Java Architecture

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

Compilation of code

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

A Servlet is a Java class, which conforms to the Java Servlet API, a protocol by which a Java class may respond to http requests. Thus, a software developer may use a servlet to add dynamic content to a Web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlets are the Java counterpart to non-Java dynamic Web content technologies such as CGI and ASP.NET. Servlets can maintain state in session variables across many server transactions by using HTTP cookies, or URL rewriting.

The servlet API, contained in the Java package hierarchy javax.servlet, defines the expected interactions of a Web container and a servlet. A Web container is essentially the component of a Web server that interacts with the servlets. The Web container is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights.

A Servlet is an object that receives a request and generates a response based on that request. The basic servlet package defines Java objects to represent servlet requests and responses, as well as objects to reflect the servlet's configuration parameters and execution environment. The package javax.servlet.http defines HTTP-specific subclasses of the generic servlet elements, including session management objects that track multiple requests and responses between the Web server and a client. Servlets may be packaged in a WAR file as a Web application.

ORACLE:Oracle Database 10g Express Edition exceeds dependability requirements and provides innovative capabilities that increase employee effectiveness, Integrate heterogeneous IT ecosystems, and maximize capital and operating budgets. Oracle Database 10g Express Edition provides the enterprise data management platform our organization needs to adapt quickly in a fast-changing environment.With the lowest implementation and maintenance costs in the industry, Oracle Database 10g Express Edition delivers rapid return on the data management investment. Oracle Database 10g Express Edition supports the rapid development of enterprise-class business applications that can give our company a critical competitive advantage.

Benchmarked for scalability, speed, and performance, Oracle Database 10g Express Edition is a fully enterprise-class database product, providing core support for Extensible Markup Language (XML) and Internet queries.

JAVASCRIPT: JavaScript is a script-based programming language that was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java. JavaScript supports the development of both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write Web server programs that can process information submitted by a Web browser and then updates the browser's display accordingly.

IV. IMPLEMENTATION OF SYSTEM

JDBC ARDHITECTURE

The JDBC API uses a Driver Manager and database-specific drivers to provide transparent connectivity to heterogeneous databases. The location of the driver manager with respect to the JDBC drivers and the servlet is shown in

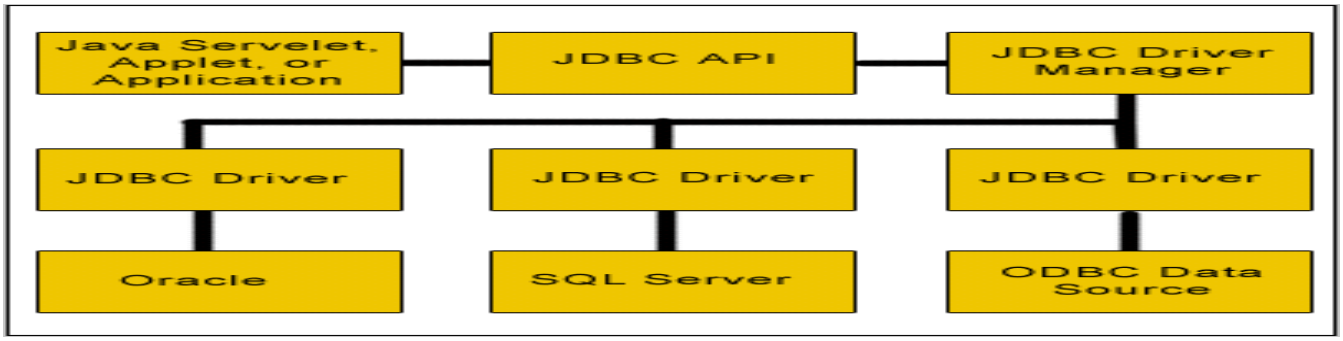


Figure.

Layers of the JDBC Architecture

. And finally the query is executed by the database. This driver has serious limitation for many applicationsA JDBC driver translates standard JDBC calls into a network or database protocol or into a database library API call that facilitates communication with the database. This translation layer provides JDBC applications with database independence. If the back-end database changes, only the JDBC driver need be replaced with few code modifications required. There are four distinct types of JDBC drivers

JDBC Driver and Its Types

Type 1 JDBC-ODBC Bridge.

Type 1 drivers act as a *"bridge"* between JDBC and another database connectivity mechanism such as ODBC. The JDBC- ODBC Bridge provides JDBC access using most standard ODBC drivers. This driver is included in the Java 2 SDK within the sun.jdbc.odbc package. In this driver the java statements are converted to jdbc statements. A JDBC statement calls the ODBC by using the JDBC-ODBC Bridge

Code for NYMBLE Ticket generation:

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<html>
<head>
<title>NYMBLEMANAGER PAGE</title>
<style type="text/css">
body {
height: 100%;
background:url("./images/g.jpg");
background-repeat:no-repeat;
text-decoration:none;
}
</style>
<script language="javascript">
function noback()
{
window.history.forward();
}
</script>
</head>
<body onLoad="noback()">
<center>
<font size="28" color="green">
WELCOME TO NYMBLE MANAGER
</font>
</center>
<%String username=(String)session.getAttribute("username");
session.setAttribute("username",username);%>
Welcome to <%=username%>
<%
HttpSession s=request.getSession(true);
```



```
String r=s.getAttribute("ticket").toString();
%>
<center>
<h1>Your Generated Nymble ticket is:
<%=r%></h1></center>
<%
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","nymble","nymble");
out.println("connected");
Statement st=con.createStatement();
st.executeQuery("select username from register where username='"+username+"'");
%>
<center>
<form name="f4" action="userhome.jsp">
<fieldset>
<label>
<h3>
Enter your generated ticket code here:
<input type="text" name="tick">
<br><br>
<br>
<input type="submit" value="submit">
</h3>
</label>
</fieldset>  </form> </center>
</body>
</html>
```

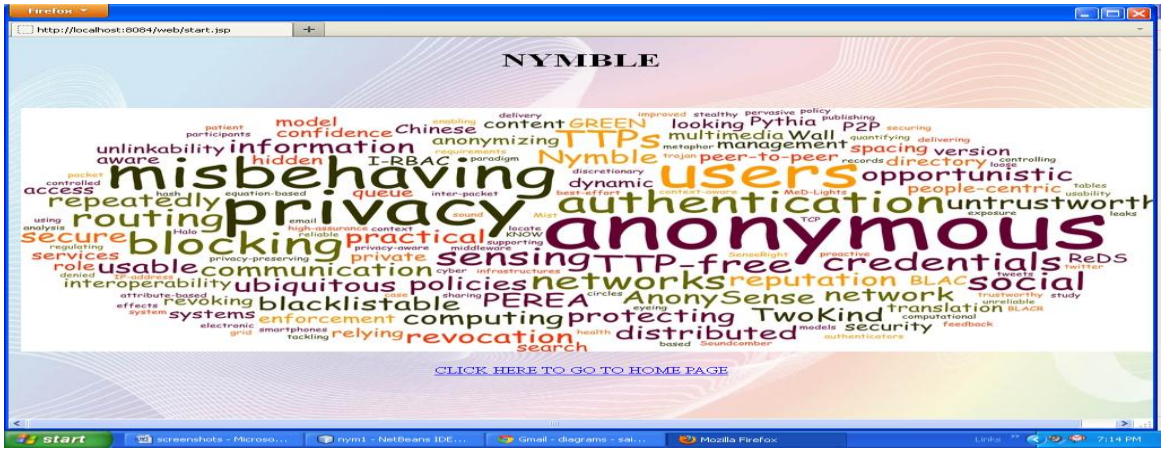
Code for blacklisting the user:

```
<%@include file="connection.jsp"%>
<html>
<head>
<title>userhome Page</title>
<style type="text/css">
body {
height: 100%;
background:url("./images/g.jpg");
text-decoration:none;
background-repeat:no-repeat;
}
</style>
<body>
<% HttpSession ss=request.getSession(true);
String userid=ss.getAttribute("userid").toString();
Statement st=con.createStatement();
ResultSet rs=st.executeQuery("select username,useremail from register where userid='"+userid+"'");
if(rs.next())
{
String username=rs.getString(1);
String email=rs.getString("useremail");
ss.setAttribute("username", username);
ss.setAttribute("email",email);
}
%>
<%String username=(String)session.getAttribute("username");
session.setAttribute("username",username);%>
Welcome to <%=username%>
<font color="red" size="12">
<center> Welcome User</center>
</font>
```



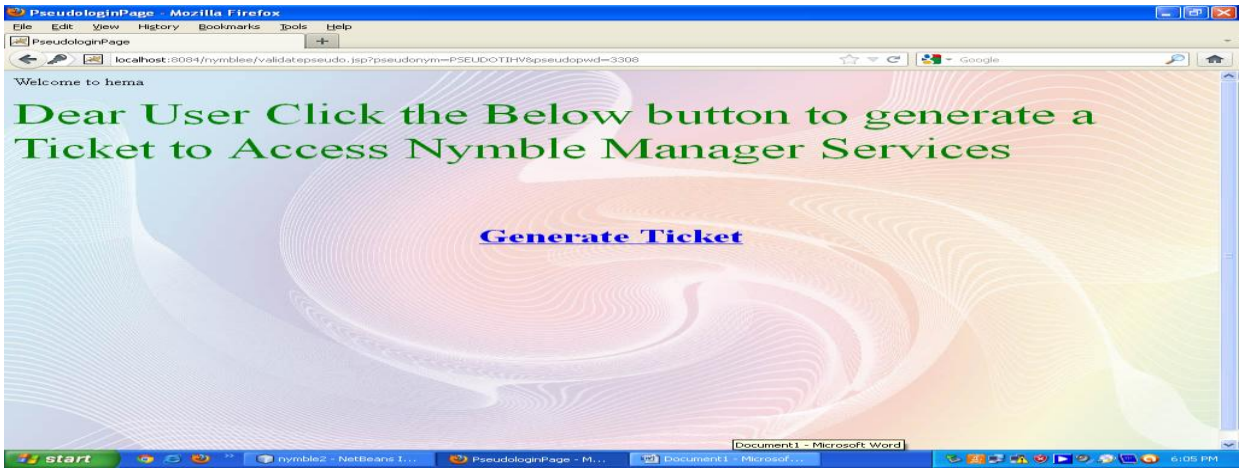
```
<br><br>
<%
String a=request.getParameter("value");
if(a!=null)
{
if(a.equals("1"))
out.println("<html><center><br><font color=red>YOU ARE BALCKLISTED</center><br><br></html>");
}
%>
<table align="center">
<tr><td colspan="30">
<a href="server.jsp"><font size="4" color="green"> SERVER ACCESS </font></a>
</td>
<td colspan="40"><a href="viewprofile.jsp"><font size="4" color="green"> View Profile </font></a></td>
<td colspan="40"><a href="request.jsp"><font size="4" color="green"> Send Request</font></a>

</td>
<td colspan="40"><a href="editprofile.jsp"><font size="4" color="green"> Edit Profile </font></a></td>
<td colspan="40"><a href="signout.jsp"><font size="4" color="green"> SignOut</font></a>
</td>
<td colspan="40"><a href="userfileupload.jsp"><font size="4" color="green"> File Upload</font></a>
</td>
<td colspan="40"><a href="FileDownloadBlock.jsp"><font size="4" color="green"> File Download</font></a>
</td>
</table>
```

V. Experimental Results



HOME PAGE

LOGIN PAGE



NYMBEL MANAGER GENERATING TICKET

USER HAS BEEN BLOCKED FOR TRYING TO UPLOAD A FILE WHICH IS NOT ALLOWED BY NYMBLE

AFTER LOGOUT:

VI. CONCLUSION

Nymble is based on two administratively-separate "manager" servers, the Pseudonym Manager (PM) and the Nymble Manager (NM). The PM is responsible for pairing a user's IP address with a pseudonym deterministically generated based on the user's IP address. The NM pairs a user's pseudonym with the target server. As long as the two managers are not

colluding, the user's connections remain anonymous to the PM, pseudonymous to the NM (note that the user does not communicate directly with the NM, and connects to the NM through Tor), and anonymous to servers that the user connects to. A user's connections within a time period are tied
to a single nymble ticket. If and when a user misbehaves, the server may not realize it for some
amount of time and may not report it until a later time period. However, after receiving a linking
token the server is able to block all future connections until the next linkability window. This is done for two reasons:

- Dynamism: IP-addresses can be reassigned to different, well-behaved users making it undesirable to permanently blacklist IP-addresses.
- Forgiveness: It ensures that bad behavior is forgiven after a certain amount of time.

Finally we conclude that nymble is a system that allows websites to selectively blacklist users of
anonymizing networks such as Tor without knowing the user's IP-address. Users not on the blacklist enjoy anonymity while blacklisted users are not allowed future connections for a
duration of time while their previous connections remain unlinkable. Since the websites to blacklist anonymious users of their choice, and since users are notified of blacklist status, Nymble gives websites the power to define their own definition of "misbehavior". Our hope is that Nymble's properties well make the usage of anonymizing networks such as Tor more acceptable.

References

[1] NYMBLE: Protecting the Privacy of Users in Anonymous Networks and Blacklisting Misbehaving Users R.Anto Arockia Rosaline, International Journal of Engineering, Business and Enterprise Applications (IJEBEA)
[2] M.Bellare, H.Shi and C.Zhang. Foundations of Group Signatures: The case of Dynamic Groups. In CT-RSA, LNCS 3376, pages 136–153. Springer, 2005.
[3] D.Boneh and H.Shacham. Group Signatures with Verifier-Local Revocation. In ACM Conference on Computer and Communications Security, pages 168–177. ACM, 2004.
[4] E.Bresson and J.Stern. Efficient Revocation In Group Signatures. In Public Key Cryptography, LNCS 1992, pages 190–206. Springer, 2001.
[5] P.Tsang, A.Kapadia, C.Cornelius and S.W.Smith. Nymble: Blocking Misbehaving Users in Anonymizing Networks. In IEEE Transactions on Dependable And Secure Computing, VOL 8, March- April 2011.
[6] A. Juels and J. G. Brainard. Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks. In NDSS.The Internet Society, 1999. Rosaline et al., International Journal of Engineering, Business and Enterprise Applications, 2 (1), Aug-Nov, 2012, pp. 26-30 IJEBEA 12-207, © 2012, IJEBEA All Rights Reserved Page 30
[7] S. Micali. NOVOMODO: Scalable Certificate Validation and Simplified PKI Management. In 1st Annual PKI Research Workshop - Proceeding, April 2002.
[8] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. PEREA: Towards Practical TTP-Free Revocation in Anonymous Authentication.In ACM Conference on Computer and Communications Security, pages 333–344. ACM, 2008.
[9] J. E. Holt and K. E. Seamons.Nym: Practical Pseudonymity for Anonymous Networks. Internet Security Research Lab Technical Report 2006-4, Brigham Young University, June 2006.

Biography:

Mr B.Srinivasulu, Post Graduated in Computer Science Engineering (M.Tech) From JNTUH,2010, and graduated in Computer Science Engineering (B.Tech)from JNTU Hyderabad, 2008. He is working presently as Assitant Professor in Department of Computer Science & Engineering in St.Martin's Engineering College, RR Dist, A.P, INDIA. He is has 3+ years Experience.His Research Interests Include Software Engineering, Network Security & Cloud Computing.

Ms SukanyaSripathi, Post Graduated in Computer Science (M.Tech), JNTUH, 2012, and Graduated in Computer Science & Engineering (B.Tech) From JNTU Hyderabad, 2007.He is working presently as AssitantProfessor in Department of Computer Science & Engineering in Brilliant Institute of Engineering& Technology, RR Dist, A.P, INDIA. She has 3+ years Experience. Her Research Interests Include Software Engineering, Cloud Computing, Operating Systems and Information Security.

Mr A.Bhima Sanakram, Post Graduated in Computer Science & Technology (M.Tech), Andhra University, 2010, and graduated in Information Technology (B.Tech) from JNTU Hyderabad, 2008. He is working presently as Assistant Professor in Department of Computer Science & Engineering in St. Martin's Engineering College, RR Dist, A.P, INDIA. He is has 2+ years Experience. His Research Interests Include Software Engineering & Cloud Computing.

Mr Rayapati Venkata Sudhakar, Post Graduated in Computer Science & Engineering (M.Tech) , JNTUH , 2008, and graduated in Information Technology (B.Tech) From JNTU Hyderabad, 2005. He is working presently as Associate Professor in Department of Computer Science & Engineering in St. Martin's Engineering College, RR Dist, A.P, INDIA. He has 5+ years Experience. His Research Interests Include Software Engineering & Cloud Computing.

.

