**Chapter 1**

**Introduction**

**1.1 Scope of the project**

The Domain Name System (DNS) has become a critical operational part of the internet infrastructure, yet it has no strong security mechanisms to assure data integrity or authentication. Extensions to the DNS are described that provide these services to security aware resolves are applications through the use of Cryptographic Digital Signatures. These Digital Signatures are included zones as resource records.

The extensions also provide for the storage of authenticated public keys in the DNS. This storage of keys can support general public key distribution services as well as DNS security. These stored keys enables security aware resolvers to learn the authenticating key of zones, in addition to those for which they are initially configured. Keys associated with DNS names can be retrieved to support other protocols. In addition, the security extensions provide for the authentication of DNS protocol transactions.

The DNS Security is designed to provide security by combining the concept of both the Digital Signature and Asymmetric key (public key) Cryptography, here the public key is sent instead of private key. The DNS security uses Message Digest Algorithm to compress the message (text file) and PRNG (Pseudo Random Number Generator) algorithm for generating public and private key. The message combines with the private key to form a signature using DSA algorithm, which is sent along with the public key.

The receiver uses the public key and DSA algorithm to form a Signature. If this signature matches with the signature of the message received, the message is decrypted and read else discarded.

###### 1.2 Problem Statement

Authenticity is based on the identity of some entity. This entity has to prove that it is genuine. In many network applications the identity of participating entities is simply determined by their names or addresses. High levelapplications use mainly names for authentication purposes, because address lists are much harder to create, understand, and maintain than name lists.

Assuming an entity wants to spoof the identity of some other entity, it is enough to change the mapping between its low level address and its high level name. It means that an attacker can fake the name of someone by modifying the association of his address from his own name to the name he wants to impersonate. Once an attacker has done that, an authenticator can no longer distinguish between the true and fake entity.

**Chapter 2**

**Overview of the DNS**

To connect to a system that supports IP, the host initiating the connection must know in advance the IP address of the remote system. An IP address is a 32-bit number that represents the location of the system on a network. The 32-bit address is separated into four octets and each octet is typically represented by a decimal number. The four decimal numbers are separated from each other by a dot character (".").The Stanford Research Institute’s Network Information Center (SRI-NIC) became the responsible authority for maintaining unique host names for the Internet. The SRI-NIC maintained a single file, called hosts.txt, and sites would continuously update SRI-NIC with their host name to IP address mappings to add to, delete from, or change in the file. The problem was that as the Internet grew rapidly, so did the file causing it to become increasingly difficult to manage. Moreover, the host names needed to be unique throughout the worldwide Internet. With the growing size of the Internet it became more and more impractical to guarantee the uniqueness of a host name. The need for such things as a hierarchical naming structure and distributed management of host names paved the way for the creation of a new networking protocol that was flexible enough for use on a global scale [ALIU].

What evolved from this is an Internet distributed database that maps the names of computer systems to their respective numerical IP network address(es). This Internet lookup facility is the DNS. Important to the concept of the distributed database is delegation of authority. No longer is one single organization responsible for host name to IP address mappings, but rather those sites that are responsible for maintaining host names for their organization(s) can now regain that control.

**2.1 Fundamentals of DNS**

The DNS not only supports host name to network address resolution, known as forward resolution, but it also supports network address to host name resolution, known as inverse resolution. Due to its ability to map human memorable system names into computer network numerical addresses, its distributed nature, and its robustness, the DNS has evolved into a critical component of the internet, without it, the only way to reach other computers on the internet is to use the numerical network address. Using IP addresses to connect to remote computer systems is not a very user-friendly representation of a system’s location on the Internet and thus the DNS is heavily relied upon to retrieve an IP address by just referencing a computer system's Fully Qualified Domain Name (FQDN). A FQDN is basically a DNS host name and it represents where to resolve this host name within the DNS hierarchy.

**2.1.1 The Domain Name Space**

The DNS is a hierarchical tree structure whose root node is known as the root domain. A label in a DNS name directly corresponds with a node in the DNS tree structure. A label is an alphanumeric string that uniquely identifies that node from its brothers. Labels are connected together with a dot notation, ".", and a DNS name containing multiple labels represents its path along the tree to the root. Labels are written from left to right. Only one zero length label is allowed and is reserved for the root of the tree. This is commonly referred to as the root zone. Due to the root label being zero length, all FQDNs end in a dot [RFC 1034].

|  |
| --- |
| As a tree is traversed in an ascending manner (i.e., from the leaf nodes to the root), the nodes become increasingly less specific (i.e., the leftmost label is most specific and the right most label is least specific). Typically in an FQDN, the left most label is the host name, while the next label to the right is the local domain to which the host belongs. The local domain can be a subdomain of another domain. The name of the parent domain is then the next label to the right of the subdomain (i.e., local domain) name label, and so on, till the root of the tree is reached |

**Figure 1.  Domain Name Space example**

When the DNS is used to map an IP address back into a host name (i.e., inverse resolution), the DNS makes use of the same notion of labels from left to right (i.e., most specific to least specific) when writing the IP address. This is in contrast to the typical representation of an IP address whose dotted decimal notation from left to right is least specific to most specific. To handle this, IP addresses in the DNS are typically represented in reverse order. IP addresses fall under a special DNS top level domain (TLD), known as the in-addr.arpa domain. By doing this, using IP addresses to find DNS host names are handled just like DNS host name lookups to find IP addresses.

|  |
| --- |
|  |

**Figure 2.  Example of inverse domains and the Domain Name Space**

**2.1.2 DNS Components**

The DNS has three major components,

* + The database
  + The server
  + The client

The database is a distributed database and is comprised of the Domain Name Space, which is essentially the DNS tree, and the Resource Records (RRs) that define the domain names within the Domain Name Space. The server is commonly referred to as a name server. Name servers are typically responsible for managing some portion of the Domain Name Space and for assisting clients in finding information within the DNS tree. Name servers are authoritative for the domains in which they are responsible. They can also serve as a delegation point to identify other name servers that have authority over subdomains within a given domain.

**2.2 Proposed System**

Taking the above prevailing system into consideration the best solution is using Pseudo Random Number Generator for generating key pair in a quick and more secured manner. We use MD5 (or) SHA-1 for producing MessageDigest and compressing the message. Signature is created using private key and MessageDigest which is transmitted along with the Public Key. The transfer of the packets from each system to system is shown using Graphical User Interface (GUI). Each time the System get the message, it verifies the IP address of the sender and if no match is found it discards it. For verification, the destination system generates signature using public key and DSA algorithm and verifies it with received one. If it matches it decrypts otherwise it discards.

The Following functions avoids the pitfalls of the existing system.

* Fast and efficient work
* Ease of access to system
* Manual effort is reduced

**Chapter 3**

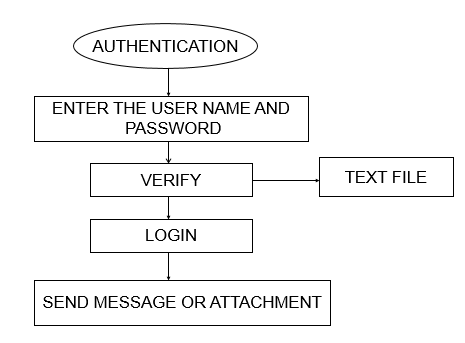
**Future Implementation Steps**

The following steps will be followed in the implementation of the proposed system:-

* Authentication
* Message Encryption using Message Digest Algorithm
* Key Generation using PRNG Algorithm
* Signature Generation
* Verifying Signature and Decrypting

**Step 1: Authentication**

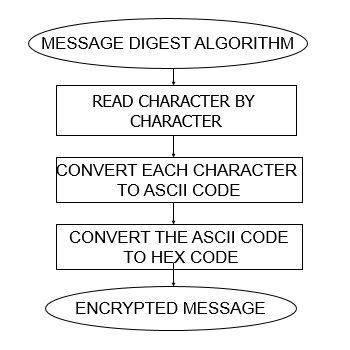
The following flowchart describes the authentication process.

****

**Figure 3. Authentication Process**

**Step 2: Message Encryption using Message Digest Algorithm**

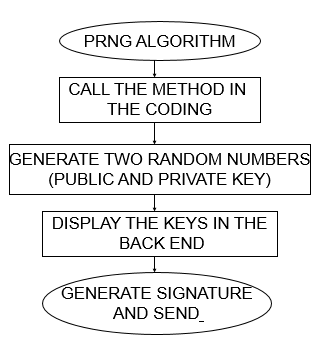
The following flowchart describes the message encryption process.

****

**Figure 4. Message Encryption Process**

**Step 3: Key Generation using PRNG Algorithm**

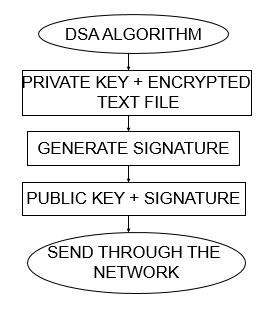
The following flowchart describes the key generation process.

****

**Figure 5. Key Generation Process**

**Step 4: Signature Generation**

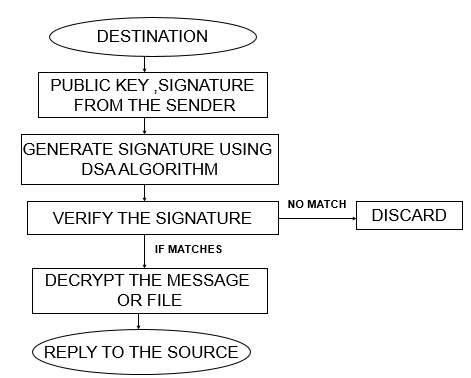
The following flowchart describes the signature generation process.

****

**Figure 6. Signature Generation Process**

**Step 5: Verifying Signature and Decrypting**

The following flowchart describes the verification of signature and decryption process.

****

**Figure 7. Verifying Signature & Decryption Process**

**Chapter 4**

**JAVA Swings**

Swing provides a Graphical User Interface (GUI) to the java environment. Swing API is set of extensile GUI components which helps to create JAVA based front end/GUI applications. Swing is built on top of the AWT API and provides almost every functionality corresponding to AWT. Swing components follow a Model-View-Controller architecture.

**4.1 Swing features**

JAVA swing has the following features:-

* **Rich Controls –** Apart from the basic controls that the AWT provides, JAVA swing also provides a rich set of advanced controls like Tree, TabbedPane, slider, colorpicker, table controls.
* **Pluggable look and feel – S**wing based GUI application’s look and feel can be changed at run time based on available values.
* **Light Weight –** Swing component are independent of native Operating System's API as Swing API controls are rendered mostly using pure JAVA code instead of underlying operating system calls.
* **Highly Customizable –** Swing controls can be customized in very easy way as visual appearance is independent of internal representation.

**4.2 Swing Controls**

Every user interface considers the following three main aspects:

* **UI elements** : These are the core visual elements that the user eventually sees and interacts with.
* **Layouts:** They define how UI elements should be organized on the screen and provide a final look and feel to the GUI (Graphical User Interface).
* **Behavior:** These are events which occur when the user interacts with UI elements.

**4.2.1 Swing UI Elements**

Following is the list of commonly used controls while designed GUI using swing.

|  |  |
| --- | --- |
| **Sr. No.** | **Control & Description** |
| 1 | [**JLabel**](http://www.tutorialspoint.com/swing/swing_jlabel.htm)  A JLabel object is a component for placing text in a container. |
| 2 | [**JButton**](http://www.tutorialspoint.com/swing/swing_jbutton.htm)  This class creates a labeled button. |
| 3 | [**JCheck Box**](http://www.tutorialspoint.com/swing/swing_jcheckbox.htm)  A JCheckBox is a graphical component that can be in either an **on** (true) or **off** (false) state. |
| 4 | [**JRadioButton**](http://www.tutorialspoint.com/swing/swing_jradiobutton.htm)  The JRadioButton class is a graphical component that can be in either an**on** (true) or **off** (false) state in a group. |
| 5 | [**JList**](http://www.tutorialspoint.com/swing/swing_jlist.htm)  A JList component presents the user with a scrolling list of text items. |
| 6 | [**JTextField**](http://www.tutorialspoint.com/swing/swing_jtextfield.htm)  A JTextField object is a text component that allows for the editing of a single line of text. |
| 7 | [**JTextArea**](http://www.tutorialspoint.com/swing/swing_jtextarea.htm)  A JTextArea object is a text component that allows for the editing of a multiple lines of text. |

|  |  |
| --- | --- |
| 8 | [**ImageIcon**](http://www.tutorialspoint.com/swing/swing_imageicon.htm)  A ImageIcon control is an implementation of the Icon interface that paints Icons from Images |
| 9 | [**JScrollbar**](http://www.tutorialspoint.com/swing/swing_jscrollbar.htm)  A Scrollbar control represents a scroll bar component in order to enable user to select from range of values. |
| 10 | [**JOptionPane**](http://www.tutorialspoint.com/swing/swing_joptionpane.htm)  JOptionPane provides set of standard dialog boxes that prompt users for a value or informs them of something. |

**Table 1. Swing UI Elements**

**4.2.2 Swing Layouts**

Layout refers to the arrangement of components within the container.

**Layout Managers**

A layout manager automatically arranges your controls within a window by using some type of algorithm. Each Container object has a layout manager associated with it. A layout manager is an instance of any class that implements the LayoutManager interface. The layout manager is set by the setLayout( ) method. If no call to setLayout( ) is made, then the default layout manager is used. Whenever a container is resized (or sized for the first time), the layout manager is used to position each of the components within it.

The setLayout( ) method has the following general form:

*void setLayout(LayoutManager layoutObj)*

Here, layoutObj is a reference to the desired layout manager.

Following is the list of commonly used controls while designing GUI using swing:-

|  |  |
| --- | --- |
| **Sr. No.** | **Layout Manager & Description** |
| 1 | [**BorderLayout**](http://www.tutorialspoint.com/swing/swing_borderlayout.htm)  The borderlayout arranges the components to fit in the five regions: east, west, north, south and center. |
| 2 | [**CardLayout**](http://www.tutorialspoint.com/swing/swing_cardlayout.htm)  The CardLayout object treats each component in the container as a card. Only one card is visible at a time. |
| 3 | [**FlowLayout**](http://www.tutorialspoint.com/swing/swing_flowlayout.htm)  The FlowLayout is the default layout.It layouts the components in a directional flow. |
| 4 | [**GridLayout**](http://www.tutorialspoint.com/swing/swing_gridlayout.htm)  The GridLayout manages the components in form of a rectangular grid. |
| 5 | [**GridBagLayout**](http://www.tutorialspoint.com/swing/swing_gridbaglayout.htm)  This is the most flexible layout manager class.The object of GridBagLayout aligns the component vertically,horizontally or along their baseline without requiring the components of same size. |
| 6 | [**GroupLayout**](http://www.tutorialspoint.com/swing/swing_grouplayout.htm)  The GroupLayout hierarchically groups components in order to position them in a Container. |
| 7 | [**SpringLayout**](http://www.tutorialspoint.com/swing/swing_springlayout.htm)  A SpringLayout positions the children of its associated container according to a set of constraints. |

**Table 2. Layout Managers**

**4.2.3 Swing Event Handling**

Change in the state of an object is known as event. There are two type of events, foreground events and background events. Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism have the code which is known as event handler that is executed when an event occurs.

**Event Listeners**

The Event listeners are the interfaces responsible to handle events. Every method of an event listener method has a single argument as an object which is subclass of EventObject class.

**Event Adapters**

Adapters are abstract classes for receiving various events. The methods in these classes are empty. These classes exists as convenience for creating listener objects.

# Chapter 5

**Development Environment**

# 5.1 Hardware Environment

The minimum configuration required to run this project are:

|  |  |
| --- | --- |
| Main processor | Pentium III (or) IV |
| RAM | 128 MB |
| Hard Disk | 4.2 GB |
| Clock Speed | 550 MHz |
| System Bus Speed | 400 MHz |
| Cache RAM | 256 KB |

# Table 3. Hardware Requirements

# 5.2 Software Environment

|  |  |
| --- | --- |
| Language | JDK1.7 (or) Higher |
| Front End Design | Swings |
| Operating System | Windows |

**Table 4. Software Requirements**

# Chapter 6

**Testing**

Software Testing is a process of executing program within the intent of finding an error. Software testing is a critical element of software quality assurance and represents the ultimate review of system specification, design, coding. Testing is last chance to uncover the error defects in the software and facilities delivery of quality system,

**6.1 System Testing Requirements**

Software testing is not an activity to take up when the product is ready. An effective testing begins with a proper plan forms the user requirement stage itself. Software testability is the case with which a computer program is tested. Metrics can be used to measure the testability of a product.

**6.2 Phases of testing**

Several testing strategies and lead to the following generic characteristics:

* Testing begins then unit level and works “outward” toward the integration of the entire system.
* Different testing techniques are appropriate at different points of software development cycle

**6.2.1 Unit Testing**

In this testing all the modules of the application are tested independently.

**6.2.2 Integration Testing**

In this testing the all the modules are tested together and the interaction between them is checked.

**6.2.3 System Testing**

After the Integration testing gets over the system has a whole is tested for validation. Here the testing is done by a complete tour of all the modules in a sequence.

**6.2.4 Performance Testing**

In this testing it will be checked if the system user friendly and has a good user interface. Every user who needs to access this system will be given a user Id and password and no one else will be able to access. This too will be tested. It will be tested whether the loading of the screens of the application is fast and the migration from one form to another takes less time. The time taken for this will be calculated. The application will be designed in such a way that it occupies less memory space; the database will also be designed in such a way that it avoids duplication of records -i.e. the database avoids redundancy in all possible ways.

**6.2.5 Validation Testing**

In validation testing, all the values entered in each and every module will be tested for correctness and validation

**Chapter 7**

**Conclusion**

The mapping or binding of IP addresses to host names became a major problem. The Domain Name System (DNS) Security is designed to provide security by combining the concept of both the Digital Signature and Asymmetric key (Public key) Cryptography using Message Digest Algorithm to compress the Message(text file) and PRNG(Pseudo Random Number Generator) Algorithm for generating Private key with which a message is combined to form a digital signature. If the Signature made by the receiver using the Public Key and DSA Algorithm matches with the encrypted Signature, the message is Decrypted otherwise discarded.

**References**

1. Computer Networks by Tanenbaum
2. William Stallings, “Cryptography and Network Security: Principals and Practice”, Prentice Hall, New Jersy.Java
3. Johannes A. Buchmann, “Introduction to Cryptography”, Springer-Verlag.
4. The Complete Reference, Fifth Edition (Paperback) by Herbert Schildt “.
5. Java Unleashed (Unleashed S.) by Michael Morrison.
6. Java 2 Bible by Justin Couch and Daniel H. Steinberg