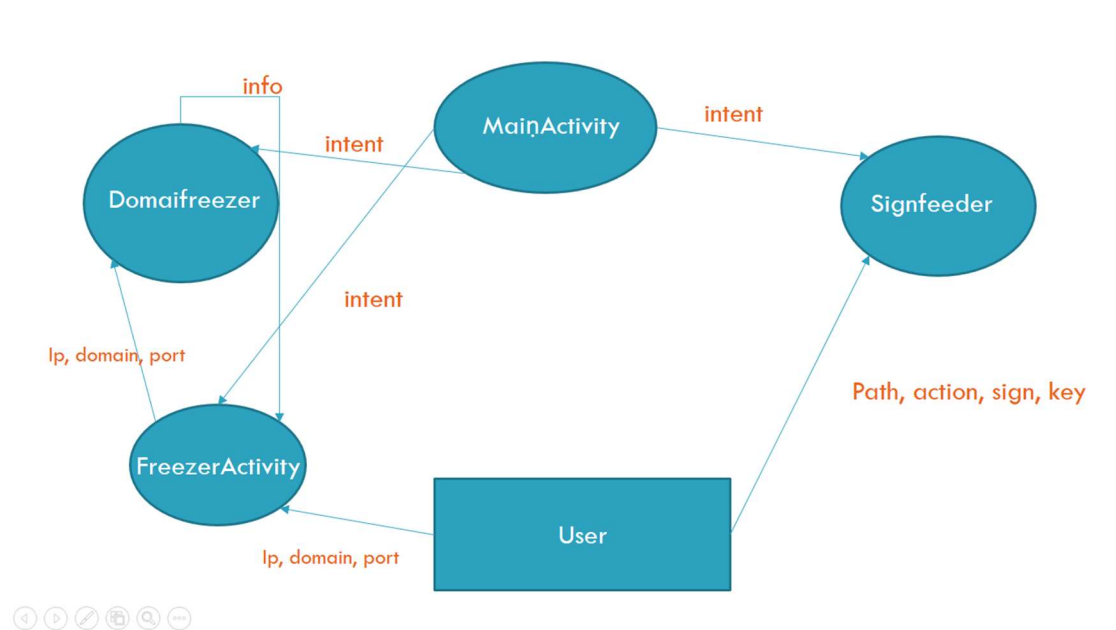


# Chapter 4

## Design

### 4.1 Data Flow Diagram



A Data Flow Diagram (DFD) shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top-down approach to Systems Design. This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modelled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.

Data flow diagrams are one of the three essential perspectives of the structured-systems analysis and design method SSADM. The sponsor of a project and the end users will

need to be briefed and consulted throughout all stages of a system's evolution. With a data flow diagram, users are able to visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's dataflow diagrams can be drawn up and compared with the new system's data flow diagrams to draw comparisons to implement a more efficient system. Data flow diagrams can be used to provide the end user with a physical idea of where the data they input ultimately has an effect upon the structure of the whole system from order to dispatch to report. How any system is developed can be determined through a data flow diagram model.

In the course of developing a set of levelled data flow diagrams, the analyst/designer is forced to address how the system may be decomposed into component sub-systems, and to identify the transaction data in the data model.

Data flow diagrams can be used in both Analysis and Design phase of the SDLC.

## **4.2 Entity Relationship Diagram**

In software engineering, an entity–relationship model (ER model) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational database. The main components of ER models are entities (things) and the relationships that can exist among them.

An entity–relationship model is the result of using a systematic process to describe and define a subject area of business data. It does not define business process; only visualize business data. The data is represented as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them, such as: one building may be divided into zero or more apartments, but one apartment can only be located in one building. Entities may have various properties (attributes) that characterize them. Diagrams created to represent these entities, attributes, and relationships graphically are called entity–relationship diagrams.

An ER model is typically implemented as a database. In the case of a relational database, which stores data in tables, every row of each table represents one instance of

an entity. Some data fields in these tables point to indexes in other tables; such pointers are the physical implementation of the relationships.

The three scheme approach to software engineering uses three levels of ER models that may be developed.

### **Conceptual data model**

This is the highest level ER model in that it contains the least granular detail but establishes the overall scope of what is to be included within the model set. The conceptual ER model normally defines master reference data entities that are commonly used by the organization. Developing an enterprise-wide conceptual ER model is useful to support documenting the data architecture for an organization.

A conceptual ER model may be used as the foundation for one or more logical data models (see below). The purpose of the conceptual ER model is then to establish structural meta data commonality for the master data entities between the set of logical ER models. The conceptual data model may be used to form commonality relationships between ER models as a basis for data model integration.

### **Logical data model**

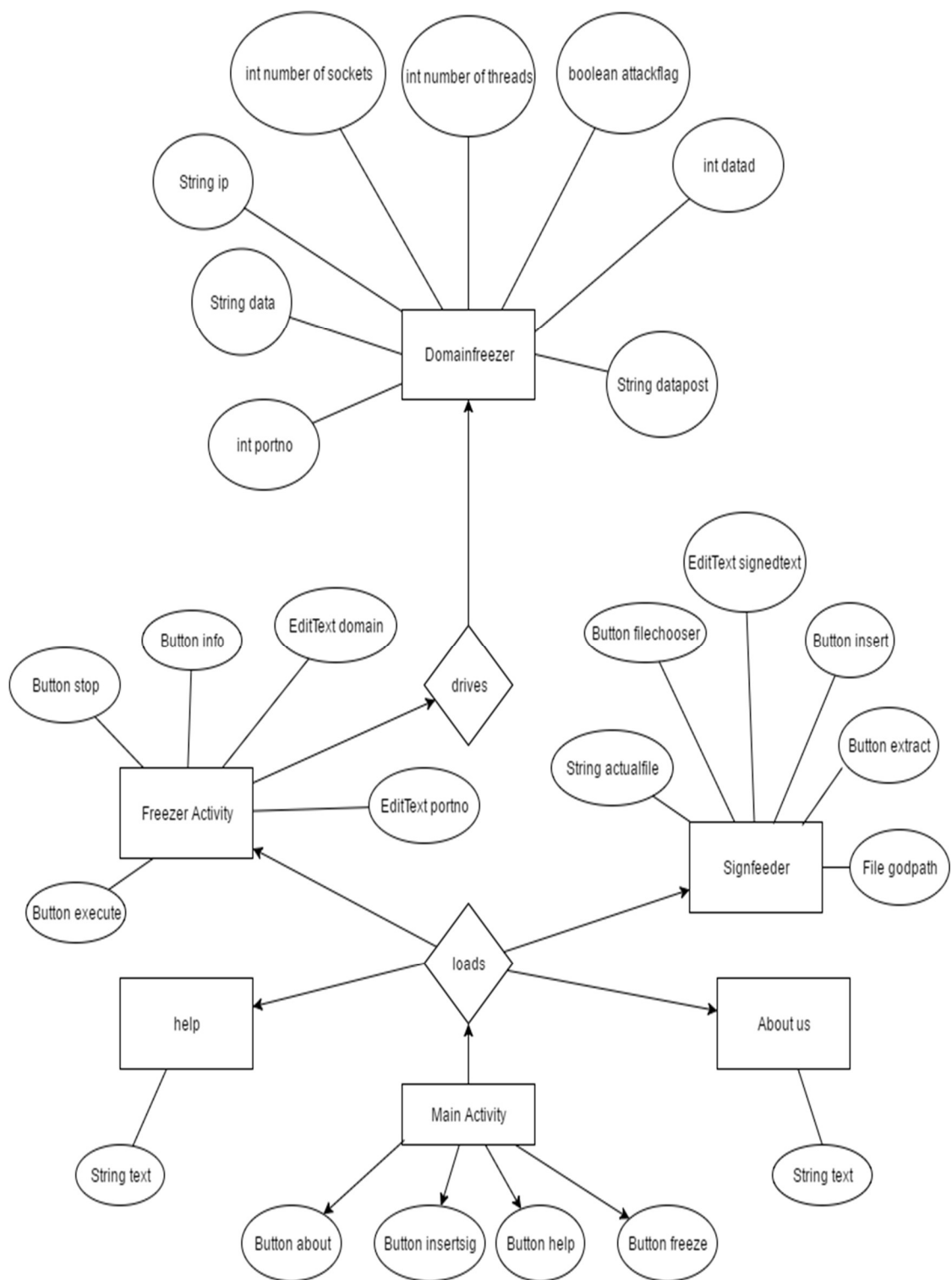
A logical ER model does not require a conceptual ER model, especially if the scope of the logical ER model includes only the development of a distinct information system. The logical ER model contains more detail than the conceptual ER model. In addition to master data entities, operational and transactional data entities are now defined. The details of each data entity are developed and the relationships between these data entities are established. The logical ER model is however developed independent of technology into which it can be implemented.

### **Physical data model**

One or more physical ER models may be developed from each logical ER model. The physical ER model is normally developed to be instantiated as a database. Therefore, each physical ER model must contain enough detail to produce a database and each physical ER model is technology dependent since each database management system is somewhat different.

The physical model is normally instantiated in the structural metadata of a database management system as relational database objects such as database tables, database indexes such as unique key indexes, and database constraints such as a foreign key constraint or a commonality constraint. The ER model is also normally used to design modifications to the relational database objects and to maintain the structural metadata of the database.

The first stage of information system design uses these models during the requirements analysis to describe information needs or the type of information that is to be stored in a database. The data modelling technique can be used to describe any ontology (i.e. an overview and classifications of used terms and their relationships) for a certain area of interest. In the case of the design of an information system that is based on a database, the conceptual data model is, at a later stage (usually called logical design), mapped to a logical data model, such as the relational model; this in turn is mapped to a physical model during physical design



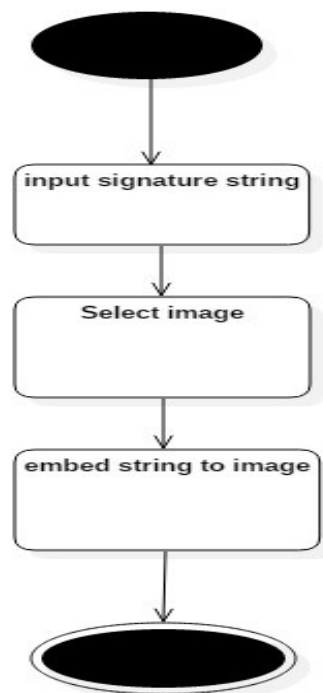
### 4.3 Flow Chart

A flowchart is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

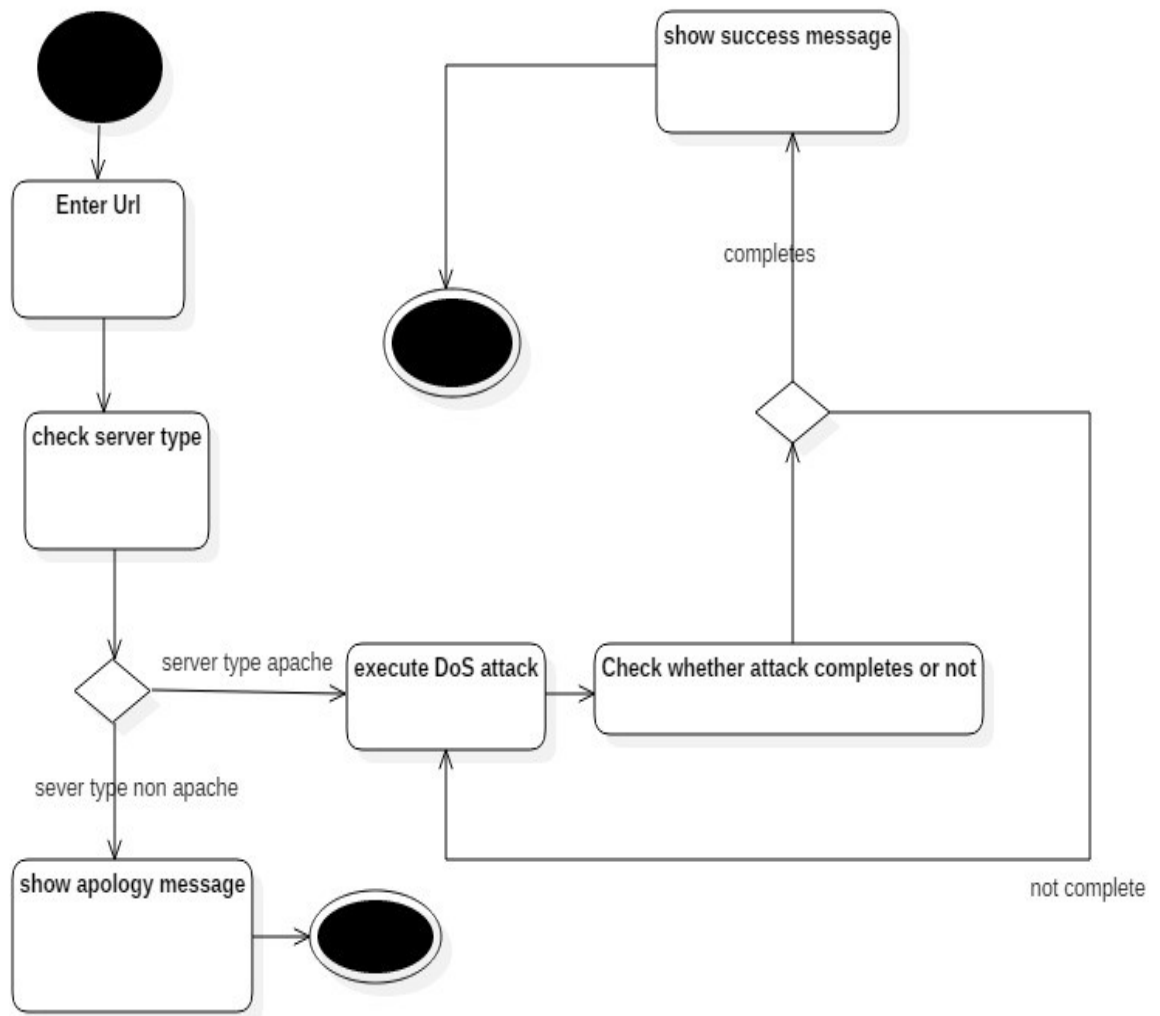
Flowcharts are used in designing and documenting simple processes or programs. Like other types of diagrams, they help visualize what is going on and thereby help understand a process, and perhaps also find flaws, bottlenecks, and other less-obvious features within it. There are many different types of flowcharts, and each type has its own repertoire of boxes and notational conventions. The two most common types of boxes in a flowchart are:

- A processing step, usually called activity, and denoted as a rectangular box.
- A decision usually denoted as a diamond.

#### 4.3.1 Flow Chart for Steganography

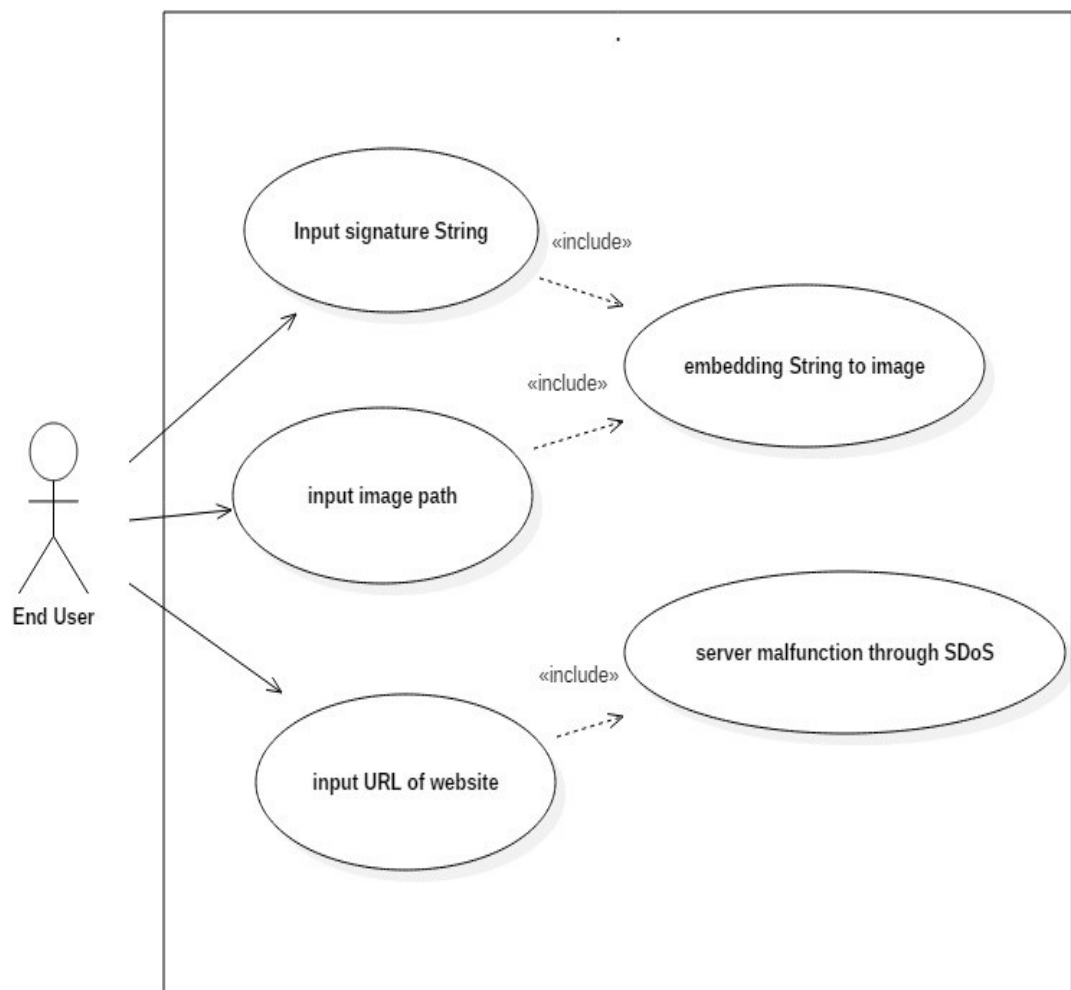


#### 4.3.2 Flow Chart for Slow Loris denial of service attack



#### 4.4 Use-Case Diagram

A use case diagram shows the relationship among actors and use cases within a system. Use case diagrams show elements from the use case model. The use case model represents functionality of a system or a class as manifested to external actors with the system.



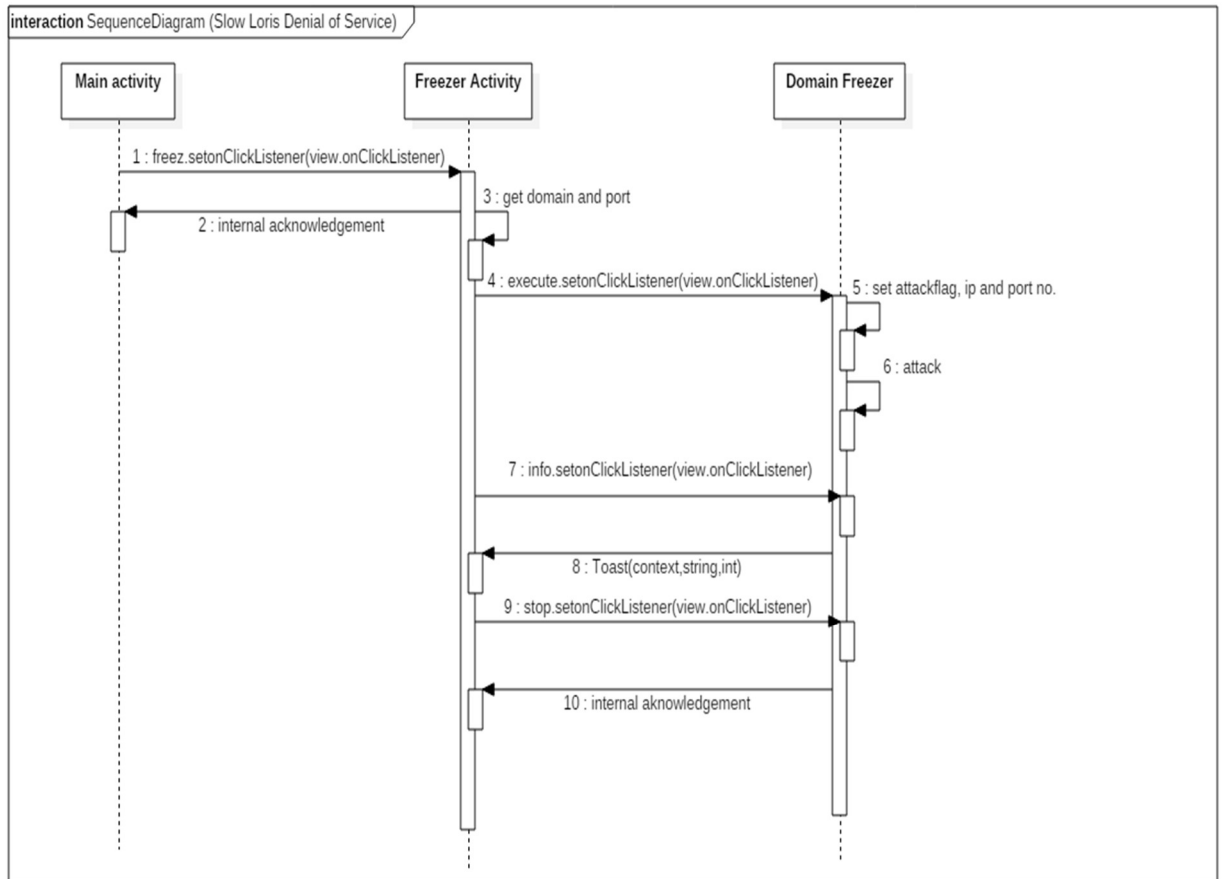
#### 4.5 Sequence Diagram

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the

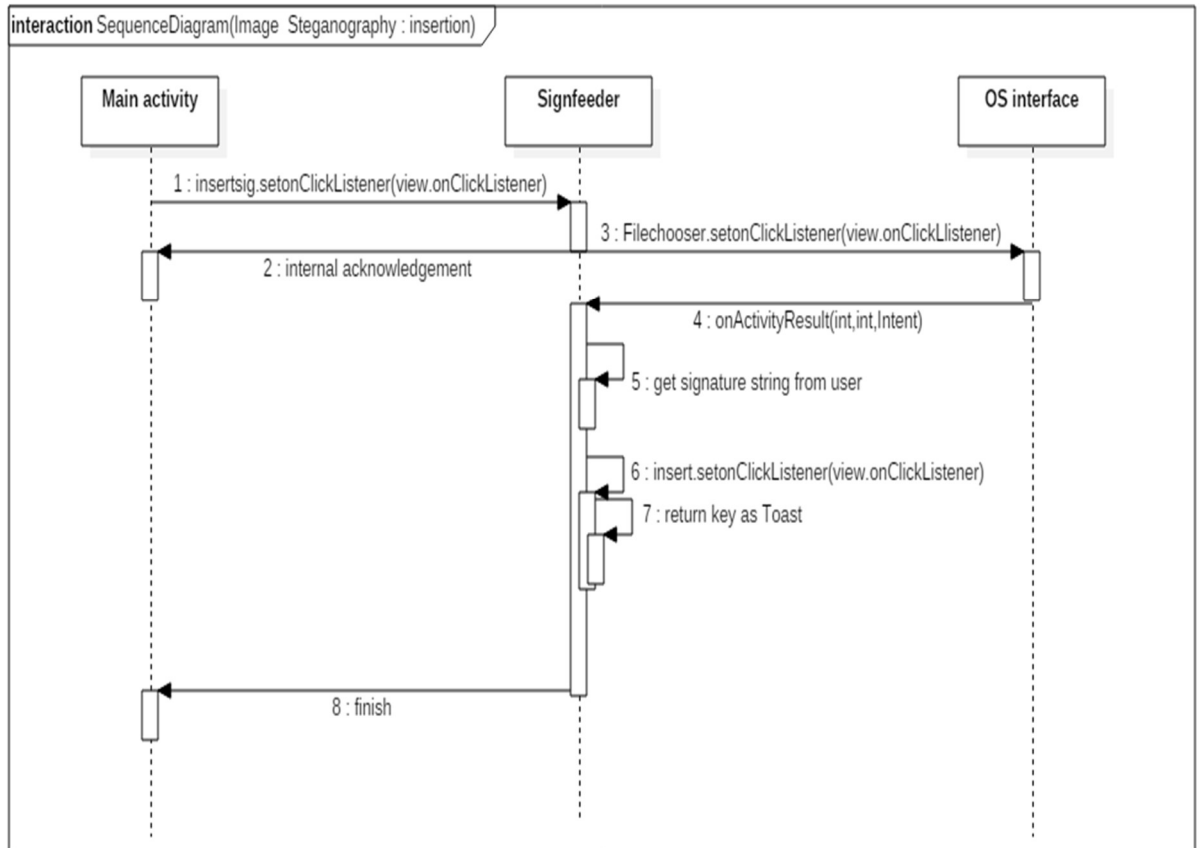


system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

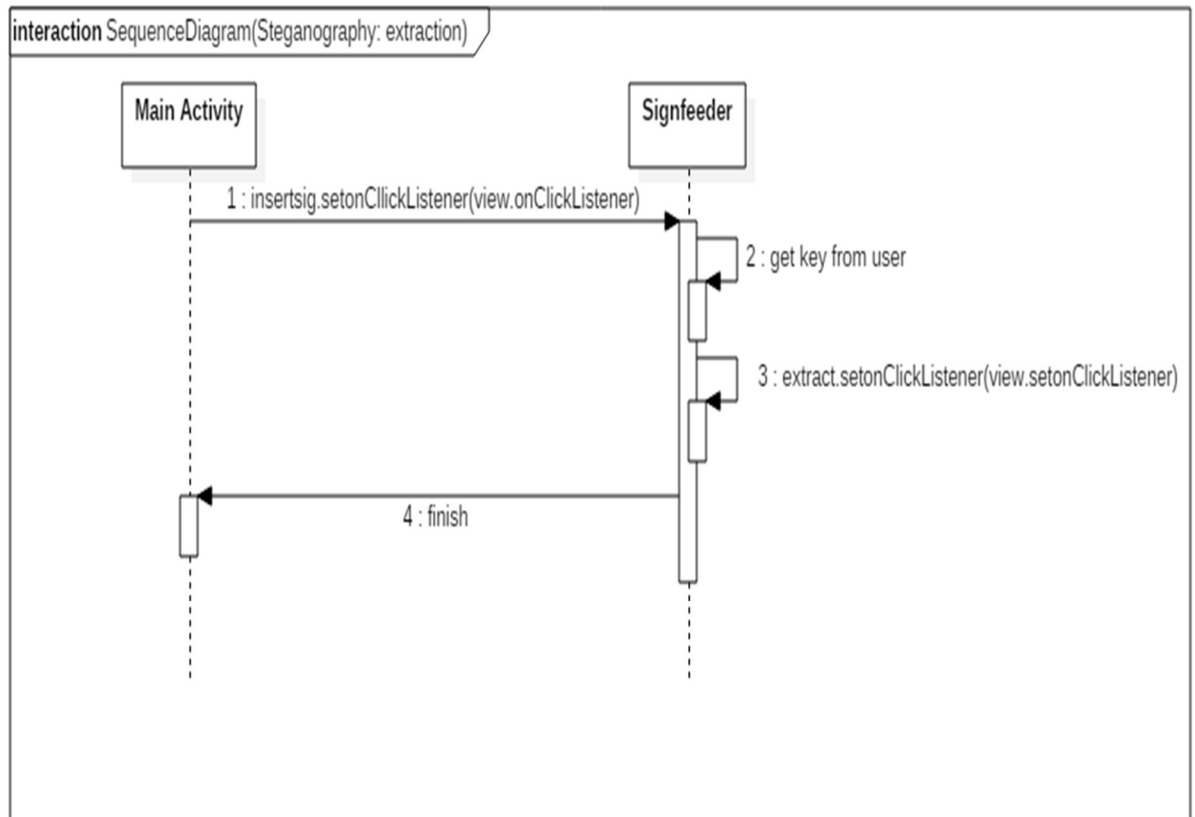
#### 4.5.1 Sequence for Slow Loris denial of service attack



#### 4.5.2 Sequence diagram for Steganography: insertion of hidden text



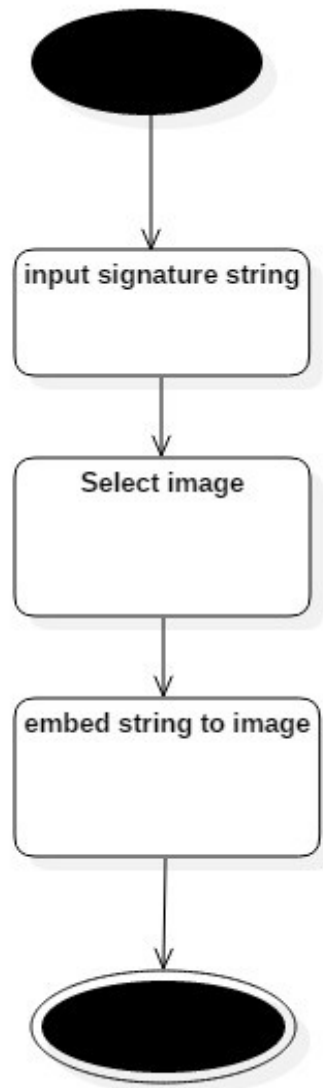
#### 4.5.3 Sequence diagram for Steganography: Extraction of hidden text



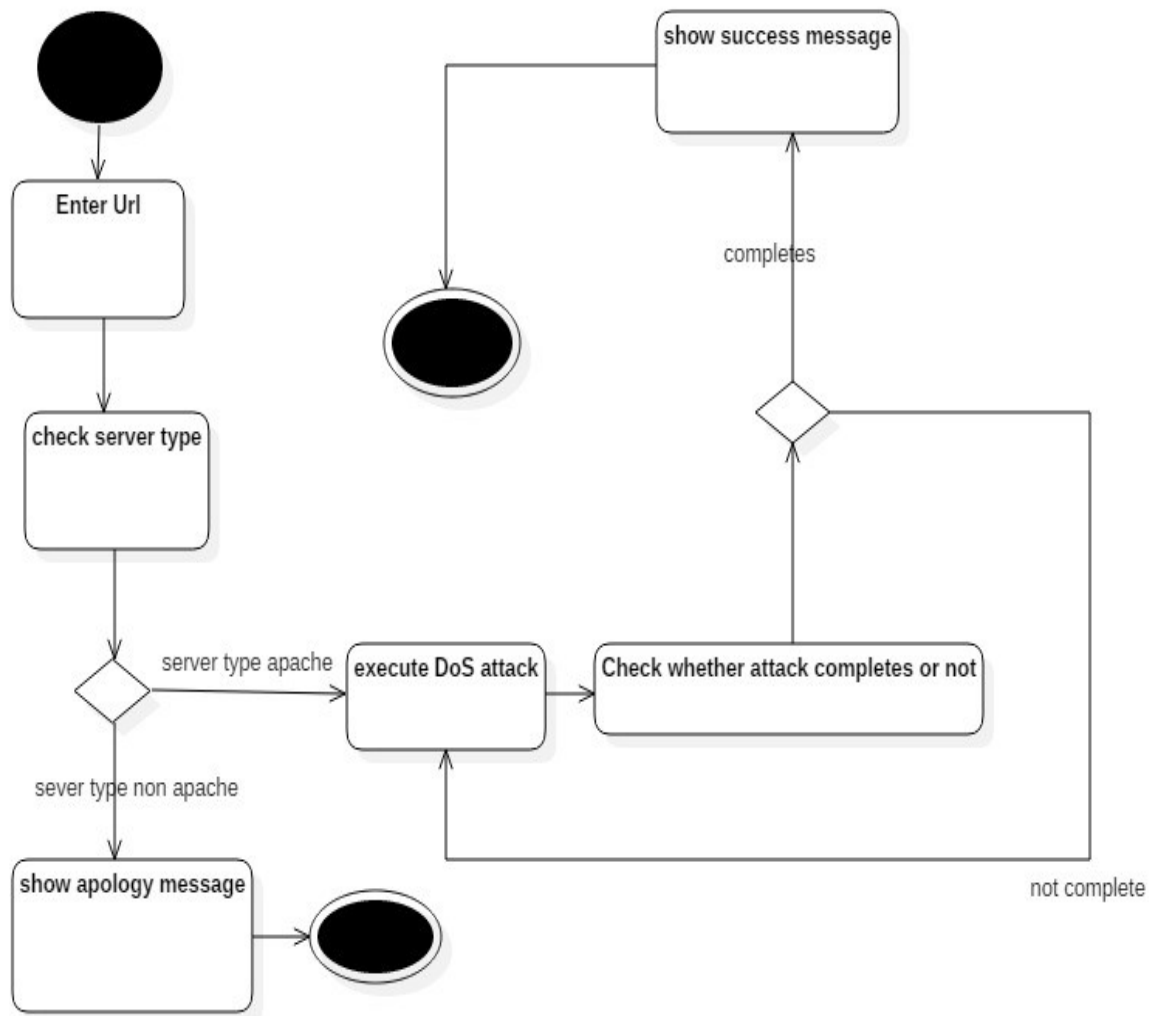
#### 4.6 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the unified modelling language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control.

#### 4.6.1 Activity Diagram for Steganography



#### 4.6.2 Activity Diagram for Slow Loris denial of service attack



## 4.7 Class Diagram

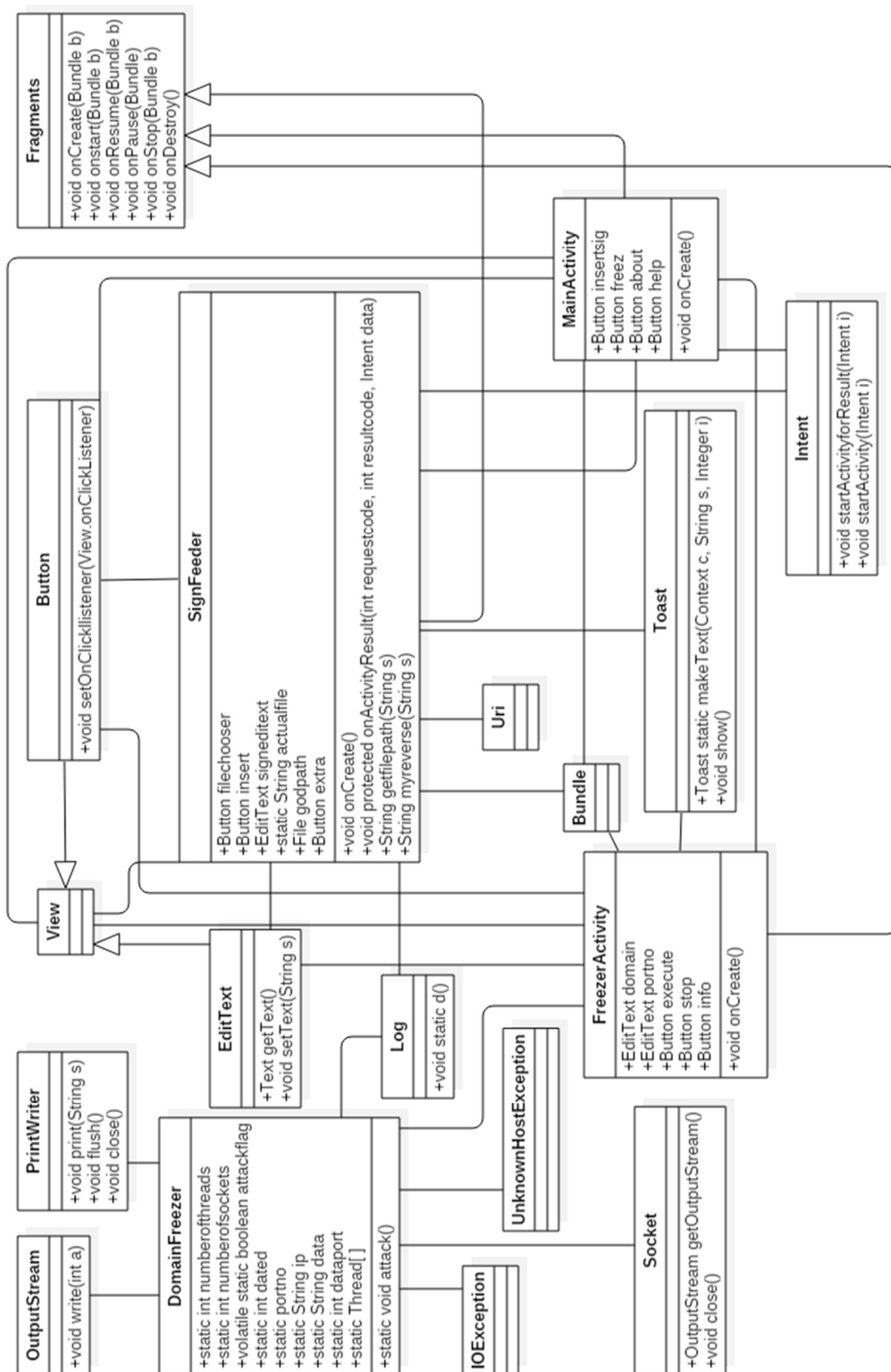
Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.

The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages. The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram.

### **Purpose:**

The purpose of the class diagram is to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction.

The UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but class diagram is a bit different. So it is the most popular UML diagram in the coder community.



## 4.8 Algorithm

An algorithm is a self-contained sequence of actions to be performed. Algorithms can perform calculation, data processing and automated reasoning tasks.

### 4.8.1 Algorithm for Steganography

#### **Begin**

Input: Cover\_Image, Secret\_Message, Secret\_Key;

Transfer Secret\_Message into Text\_File;

Zip Text\_File;

Convert Zip\_Text\_File to Binary\_Codes;

Convert Secret\_Key into Binary\_Codes;

Set BitsPerUnit to Zero;

Encode Message to Binary\_Codes;

Add by 2 unit for bitsPerUnit;

Output: Stego\_Image;

#### **End**

### 4.8.1 Algorithm for Slow Loris

#### **Begin**

Input: URL;

Create Threads of uncompleted\_HTTP\_get\_request;

#### **End**



