**PROVIDING SECURED COMMUNICATION AND BATCH VERIFICATION FOR AIRCRAFTS.**

**A PROJECT REPORT**

# *Submitted by*

**PRAVEEN KUMAR.L**

**RAJATHIRUSANGURAJ.V**

**RAMASUBRAMANIAN.S**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**



**R.M.D ENGINEERING COLLEGE, KAVAIRAIPETTAI**

**ANNA UNIVERSITY: CHENNAI 600 025**

##### **APRIL 2016**

**APPENDIX 2**

**ANNA UNIVERSITY : CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report **“PROVIDING SECURED COMMUNICATION AND BATCH VERIFICATION FOR AIRCRAFTS”** is the bonafide work of “**PRAVEEN KUMAR.L , RAJATHIRUSANGURAJ.V , RAMASUBRAMANIAN.S”** who carried out the project work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE**  **DR. P.EZHUMALAI, B.E, M.Tech, F.I.E.,Ph.D.,**  HEAD OF THE DEPARTMENT  PROFESSOR  Department of Computer Science and  Engineering,  R.M.D Engineering College,  Kavaraipettai - 601 206. | **SIGNATURE**  **Mrs. K.ROSLIN DAYANA, B.E.,M.E**  SUPERVISOR  ASSISTANT PROFESSOR  Department of Computer Science and Engineering,  R.M.D Engineering College,  Kavaraipettai - 601 206. |

Submitted for the Project viva-voice held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**External Examiner Internal Examiner**

**TABLE OF CONTENTS**

**CHAPTER TITLE PAGE NO.**

**LIST OF FIGURES ii**

**LIST OF ABBREVATIONS iii**

1. **INTRODUCTION**

1.1 About the Project 15

1. **SYSTEM ANALYSIS**

2.1 Existing system 16

2.2 Proposed system 16

**3 REQUIREMENTS SPECIFICATION**

3.1 Introduction 17

3.2 Hardware and Software specification 17

3.2.1 Hardware Requirements

3.2.2 Software Requirements

3..3 Technologies Used 18 3.3.1 Introduction to Java

3.3.2 Working of Java 19

**4 SYSTEM DESIGN**

4.1 Architecture Diagram 22

4.2 Sequence Diagram 23

4.3 Use Case Diagram 24

4.4 Activity Diagram 25

4.5 Data Base Design

**5 SYSTEM DESIGN – DETAILED**

**5.1** Modules 26

**5.2** Module explanation 26

**6 CODING AND TESTING**

**6.1** Coding 28

**6.2** Coding standards 31

**6.3** Test procedure 31

**6.4** Test data and output 32

**SNAP SHOTS**

REFERENCES 78

**ACKNOWLEDGEMENT**

We are very grateful to our Chairman **Thiru.R.S.Munirathinam** for giving us the infrastructure for conducting the project work and our Chairperson **Tmt.Manjula Munirathinam** for her blessings. We also thank our Vice Chairman, **Thiru.R.M.Kishore**, our Director, **Thiru.R.Jothi Naidu,** and our Secretary, **Thiru.Yalamanchi Pradeep** for their constant support.

We are extremely thankful to our Principal **Dr.K.Sivaram** for having given us an opportunity to serve the purpose of education. We also thank our Dean **Dr.K.Dharmalingam** and our Dean Academic **Dr.K.K.Thyagharajan** for extending their support.

We are grateful to **Dr.P.Ezhumalai,** Head of the Department of Computer Science and Engineering, for his valuable guidance and useful suggestions during the course of the project.

We are deeply thankful to our Project Coordinators for the immense and timely help. We also like to thank our project supervisor **……………………** for the support and ideas towards this project.

**ABSTRACT**

Automatic Dependent Surveillance-Broadcast (ADS-B) has become a crucial part of next generation air traffic surveillance technology and will be mandatorily deployed for most of the airspaces worldwide by 2020. Each aircraft equipped with an ADS-B device keeps broadcasting plaintext messages to other aircraft and the ground station controllers once or twice per second. The lack of security measures in ADS-B systems makes it susceptible to different attacks. Among the various security issues, we investigate the integrity and authenticity of ADS-B messages. We propose a new framework for providing ADS-B with authentication based on three-level hierarchical identity-based signature (HIBS) with batch verification. Previous signature-based ADS-B authentication protocols focus on how to generate signatures efficiently, while our schemes can also significantly reduce the verification cost, which is critical to ADS-B systems, since at any time an ADS-B receiver may receive lots of signatures. We design two concrete schemes. The basic scheme supports partial batch verification and the extended scheme provides full batch verification. We give a formal security proof for the extended scheme. Experiment results show that our schemes with batch verification are tremendously more efficient in batch verifying n signatures than verifying n signatures independently. For example, the running time of verifying 100 signatures is 502ms and 484ms for the basic scheme and the extended scheme respectively, while the time is 2500ms if verifying the signatures independently.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 4.1 | Architecture Diagram | 17 |
| 4.2 | Sequence Diagram | 18 |
| 4.3 | Use Case Diagram | 19 |
| 4.4 | Activity Diagram | 20 |
| 4.5 | Data Base Design | 21 |
| 5.2 | Patterns of the peer-peer edges | 22 |
| 5.2 | Patterns of the service-provider edges | 24 |
| 5.2 | Discovering missing links in internet |  |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **JDK** | Java Development Toolkit |
| **JMF** | Java Media Framework |
| **TCP** | Transmission Control Protocol |
| **IP** | Internet protocol |
| **HTTP** | Hyper Text Transfer Protocol |

**LIST OF SYMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | name  Class B  Class A  Class A  Class B | Associations represent static relationships between classes. Roles represent the way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single class. |
| 4 | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the process. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | Final state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 13. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 14. | External entity |  | Represents external entities such as keyboard, sensors, etc. |
| 15. | Transition |  | Represents communication that occurs between processes. |
| 16. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 17. | Message | Message | Represents the message exchanged. |
| 18. | Use case |  | Interact ion between the system and external environment. |

**1. INTRODUCTION**

**1.1 PURPOSE :**

To propose a new authentication framework for ADS-B systems supporting batch verification to ensure the integrity and authenticity of ADS-B messages.

**1.2 PROJECT SCOPE:**

Recently, a new technique will replace conventional radar systems and be deployed as part of the next generation air transportation systems. Unlike in traditional radar systems where aircraft only respond to interrogations by ground stations. In the ADS-B system, aircraft continuously obtain their positions based on some satellite positioning techniques (e.g., GPS) and periodically broadcast their positions as well as some other information such as the current velocity to ground stations and other aircraft. Recently, flight tracker Web sites based on the mashup of ADS-B data have gained popularity, providing Web users with a visual overview of air-traffic around the world.

Each aircraft equipped with an ADS-B device keeps broadcasting plaintext messages to other aircraft and the ground station controllers once or twice per second. The lack of security measures in ADS-B systems makes it susceptible to different attacks. Among the various security issues, we investigate the integrity and authenticity of ADS-B messages.

**2. SYSTEM ANALYSIS**

**2.1 EXISTING SYSTEM:**

Conventional ATC techniques are based on radar systems which include primary surveillance radars (PSR) and secondary surveillance radars (SSR). PSRs are independent and non-cooperative. Namely, PSRs transmit high-frequency signals, receive the echoes reflected from the aircraft and then can determine the position of the aircraft, without requiring the aircraft’s participation. On the other hand, SSRs cooperate with and interrogate the aircraft to get responses which are generated by the onboard systems equipped in the aircraft. The responses may contain information of the aircraft such as identification codes, height, and altitude. However, traditional PSR and SSR systems suffer from some disadvantages such as low precision and high cost.

**2.2 PROPOSED SYSTEM:**

In our proposed ADS-B system has “smart objects” that can create a graphical representation of aircraft on the Web. In this a plane object created through mashup of ADS-B data displays the basic ADS-B attributes such as call sign, registration, altitude, speed and position. Providing ADS-B data with authenticity, which is a main theme of this paper, is also important for the related Web services to maintain high level of reliability. ADS-B data are broadcast through wireless channel (radio frequency data-link) without any cryptographic mechanisms implemented. Besides the ground controllers and aircraft, anyone who holds a single low-cost ADS-B receiver can obtain the ADS-B data. We mainly deal with data integrity and source integrity. Data integrity ensures that the ADS-B data has not been modified upon arriving at the receivers. In this paper, we propose to apply hierarchical identity-based signature (HIBS) to ADS-B authentication and in fact a three-level HIBS is sufficient. The top level PKG could be an authoritative organization such as the ICAO, the FAA or EUROCONTROL. The second level consists of different airlines around the world and the aircraft stand in the third level. In addition, at any time the aircraft or the ground ADS-B receivers will receive a large number of signatures from different surrounding aircrafts which may belong to different airlines and it needs an efficient scheme to verify these signatures as soon as possible. We use batch verification to mitigate this concern. Batch verification is classified into three types, while we deal with the most intractable but desirable one, i.e. type 3. Type 3 batch verification allows multiple signatures on multiple messages generated by multiple signers to be verified at the same time batch verification can be classified into full batch verification and partial batch verification.

In this new framework, we employ a three level HIBS mechanism. The top level PKG generates private keys for the second-level PKGs (the airlines). Each airline is responsible for generating private signing keys for its affiliated aircraft. Each aircraft signs its messages with corresponding signing key and broadcasts the messages together with the signatures. And the signatures are verified in the ground station verifier (the airport).

**3. REQUIREMENT SPECIFICATIONS**

**3.1 INTRODUCTION:**

The requirements specification is a technical specification of requirements for the software products. It is the first step in the requirements analysis process it lists the requirements of a particular software system including functional, performance and security requirements. The requirements also provide usage scenarios from a user, an operational and an administrative perspective. The purpose of software requirements specification is to provide a detailed overview of the software project, its parameters and goals. This describes the project target audience and its user interface, hardware and software requirements. It defines how the client, team and audience see the project and its functionality.

**3.2 HARDWARE AND SOFTWARE SPECIFICATIONS:**

**3.2.1 HARDWARE REQUIREMENTS:**

* Hard Disk : 250GB and Above
* RAM : 2GB and Above
* Processor : Core2Duo and Above

**3.2.2 SOFTWARE REQUIREMENTS:**

* Windows operating system XP and above
* JDK 1.6
* Java FX 1.3
* Net beans 6.9
* My Eclipse 8.6

**Overall Description**

**Product Perspective**

Conventional ATC techniques are based on radar systems which include primary surveillance radars (PSR) and secondary surveillance radars (SSR). PSRs are independent and non-cooperative. Namely, PSRs transmit high-frequency signals, receive the echoes reflected from the aircraft and then can determine the position of the aircraft, without requiring the aircraft’s participation. On the other hand, SSRs cooperate with and interrogate the aircraft to get responses which are generated by the onboard systems equipped in the aircraft. The responses may contain information of the aircraft such as identification codes, height, and altitude. However, traditional PSR and SSR systems suffer from some disadvantages such as low precision and high cost.

**Product Features**

In the experiment, we run each scheme ten times to get an average value. We measure the running time of generating and verifying n signatures where n could be from 1 to 1000 increment by 100. We test for three schemes, that is: the scheme verifying n signatures independently which means verifying individual signatures in the proposed basic scheme, the basic scheme (supporting partial batch verification) and the extended scheme (support full batch verification). The result shows that generating a signature requires about 15 milliseconds (ms). The verification time is shown in Fig. 3. From the experiment results, we can see that when n > 10, both our basic scheme and extended scheme are about five times faster than verifying n signatures independently. This is because that in both the basic scheme and the extended scheme that support batch verification, the number

**User Classes and Characteristics**

* Web technology and smart objects,
* Virtual airport nodes and air craft nodes,
* Effective controlling and communications between Airports and crafts,
* Batch verification and control changes and identification of injected data.

**Keywords:**

APKG: Airline PKG

ASK: Airline Secret Key

PSK: Plane Secret Key

SKIDA: Airlines secret key

SKIDF: Aircraft secret key

**Web technology and smart object**:

Creating website to visually predict and plot our registered airports and aircrafts. The authorized elements from the root PKG (Airport authority of India) only can plot on Google map in our website. Using the highly secured networking socket communication can create the registered and authorized elements could appear on the map. The secured network application communicates through the authorized internet protocols among them. The root level main authority monitors the every occurrence of elements in the server of the website and verify with the Setup phase algorithm when the element get generated. The authorized element only gets the signatures and keys generated from the authority of root level PKG.

On input of a security parameter, the Setup algorithm allows the top PKG to generate the master secret key and the master public key. The root level PKG keeps the master key pair (msk, mpk) that will be used to generate secret keys for low-level nodes.

**Virtual airport and aircraft nodes:**

On input of the master secret key, and an airline identity IDA, the Extract A algorithm generates a secret key SKIDA for IDA. The Level-1 nodes are airlines each with an identity like IDA. The second level PKGs which can generate the secret signing keys for the bottom level nodes.

Airport nodes are created to controls and communicate with the aircraft nodes. And craft nodes show the important details like range, location, destination, craft number and the source airport of the crafts themselves. Generation of the keys and ids are based on the input of top level PKG (Authority) to the low level PKG (Airlines and aircrafts) using RSA algorithm and HMAC algorithm. Due to this attackers cannot predict the keys and signatures of the nodes.

**Effective communications and controls between Airport and Aircraft:**

On input of the secret key SKIDA for the Level-2 nodes are the aircraft each with an identity like IDF .The airline identity IDA and an aircraft identity IDF, the Extract F algorithm generates a secret key SKIDF for IDF belonging to IDA.

The messages and information’s are high sensible and secured. So it could be highly integrated and secured. On the input of SKIDF to the registered crafts are controlled and communicated by the source airports among the matched keys and signatures. And matched airports only can communicate with the aircrafts and controlled them automatically from belonged station. Information’s from the airport to aircrafts and crafts to station communications can be controlled by ground station belongs to the aircrafts. After reached the destination, above entire rolls are handled by the destination port.

**Batch verification and Control change and Identification of injected data:**

On input of the secret key SKIDF and a message m, the Sign algorithm generates a signature of m. On input of a signature on a messages m with respect to airline identity IDA and aircraft identity IDF, the Verify algorithm outputs either 0 or 1, where 1 represents that the signature is valid.

In existing system each crafts are verified individually but proposed systems implements the batch verification schemes to verify the bulky requests from the crafts. We give a formal security proof for the proposed extended scheme that supports full batch verification. Verifications are based on the SKIDF, SKIDA, signature and keys from root level PKG’s. After the verification in destination port the entire controls are handle by the destination port which is alternately changes afterwards the verification in somewhere airports.

**a) Design and Implementation Constraints**

**Constraints in Analysis**

* Constraints as Informal Text
* Constraints as Operational Restrictions
* Constraints Integrated in Existing Model Concepts
* Constraints as a Separate Concept
* Constraints Implied by the Model Structure

**Constraints in Design**

* Determination of the Involved Classes
* Determination of the Involved Objects
* Determination of the Involved Actions
* Determination of the Require Clauses
* Global actions and Constraint Realization

**Constraints in Implementation**

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.

**System Features**

* Automatic Dependent Surveillance-Broadcast (ADS-B) has become a crucial part of next generation air traffic surveillance technology and will be mandatorily deployed for most of the airspaces worldwide.
* The number of aircraft has been increasing tremendously over the last decade. So need to verify bulk number of aircrafts using Batch verification.
* We give a formal security proof for the extended scheme. Experiment results show that our schemes with batch verification are tremendously more efficient in batch verifying n signatures than verifying n signatures independently.
* With this accurate information, the ground controllers or other surrounding aircraft can monitor and track the location and path of an aircraft, which provides aircraft and the ground controllers a common situational awareness. This improves pilots’ decision-making ability dramatically and makes air traffic management much easier.

**External Interface Requirements**

**a)User Interfaces**

1. All the contents in the project are implemented using Graphical User Interface (GUI) in Java through JavaFX concepts.

2. Every conceptual part of the projects is reflected using the JavaFX.

3. System gets the input and delivers through the GUI based.

**b)Hardware Interfaces**

**Ethernet**

Ethernet on the AS/400 supports TCP/IP, Advanced Peer-to-Peer Networking (APPN) and advanced program-to-program communications (APPC).

**ISDN**

You can connect your AS/400 to an Integrated Services Digital Network (ISDN) for faster, more accurate data transmission. An ISDN is a public or private digital communications network that can support data, fax, image, and other services over the same physical interface. Also, you can use other protocols on ISDN, such as IDLC and X.25.

c) Software Interfaces

This software is interacted with the TCP/IP protocol, Socket and listening on unused ports. Server Socket and listening on unused ports and JDK 1.6

**d) Communications Interfaces**

1. TCP/IP protocol.

2. LAN settings

1. **Other Nonfunctional Requirements**
   1. **Performance Requirements**

The performance of the wireless sensor network, to execute this project on LAN or Wi-Fi communication channel. So we need to one or more than machine to execute the demo. Machine needs the enough hard disk space to install the software and run our project. 1.2 Safety Requirements

* The software may be safety-critical. If so, there are issues associated with its integrity level.
* The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.
* If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.
* There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.
* If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.
* Systems with different requirements for safety levels must be separated.
* Otherwise, the highest level of integrity required must be applied to all systems in the same environment.
  1. **Security Requirements**

Do not block the some available ports through the windows firewall.

**Software Quality Attributes**

**Functionality**: are the required functions available, including Interoperability and security.

**Reliability**: maturity, fault tolerance and recoverability

**Usability**: how easy it is to understand, learn, and operate the software System

**Efficiency**: performance and resource behavior.

**Maintainability**: Maintaining the software.

**Portability**: can the software easily be transferred to another environment, including installability.

**4. SYSTEM DESIGN**

**4.1 Architecture Diagram:**

GROUND STATION AIRPORT

(VERIFIER)

LEVEL2

PKG

(AIRCRAFT4)

LEVEL2

PKG

(AIRCRAFT3)

LEVEL2 PKG

(AIRCRAFT2)

LEVEL2

PKG

(AIRCRAFT1)

LEVEL1 PKG

(AIRPORT2)

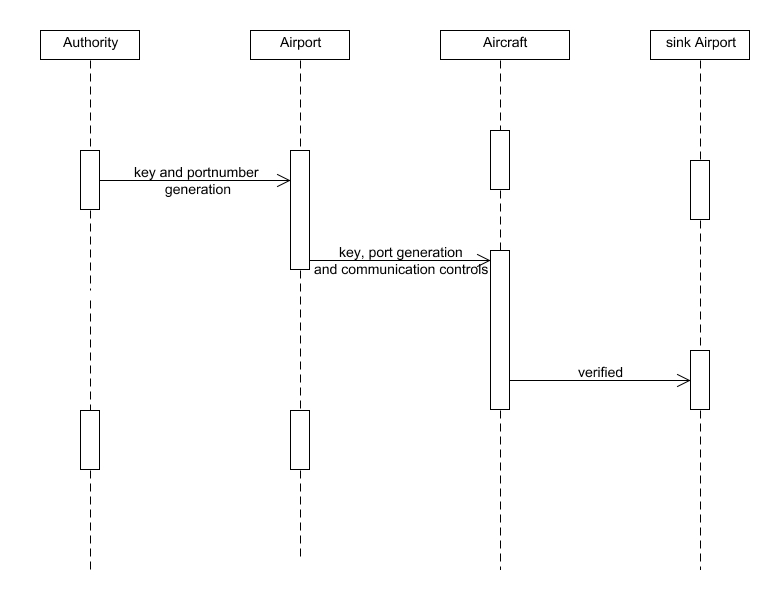
LEVEL1 PKG

(AIRPORT1)

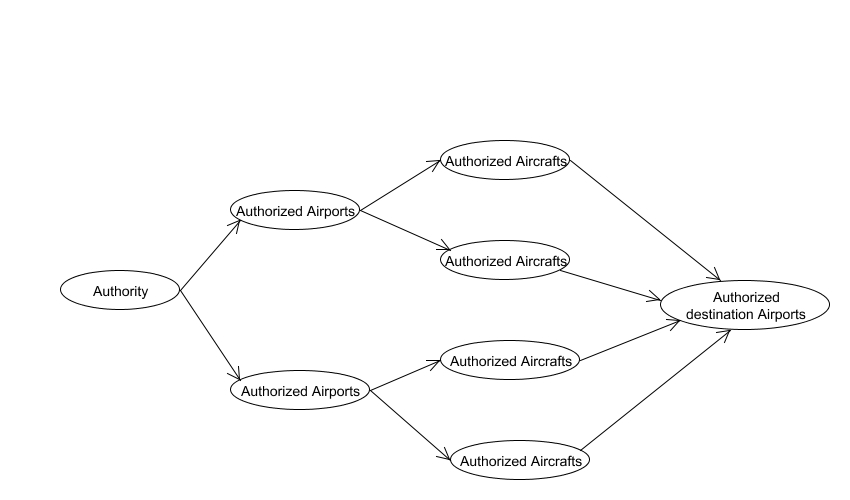
ROOT LEVEL PKG

(AIRPORT AUTHORITY)

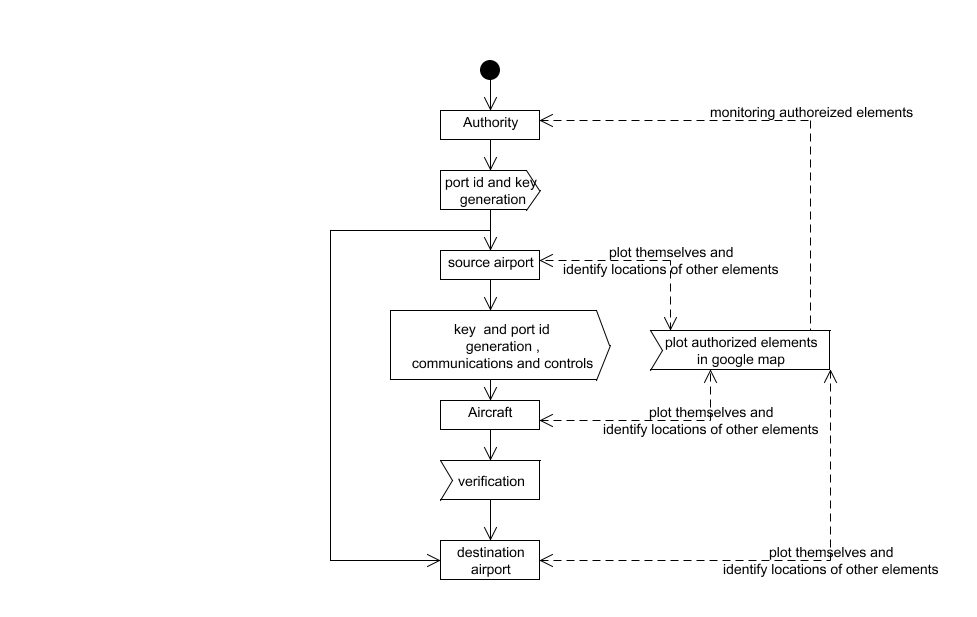
**4.2 Sequence Diagram:**

****

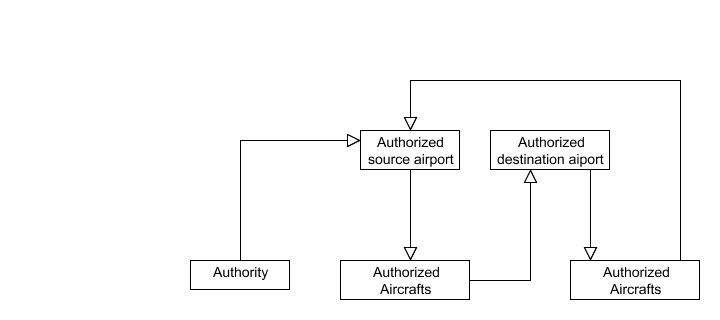
**4.3 Use Case Diagram:**

****

**4.4 Activity Diagram:**

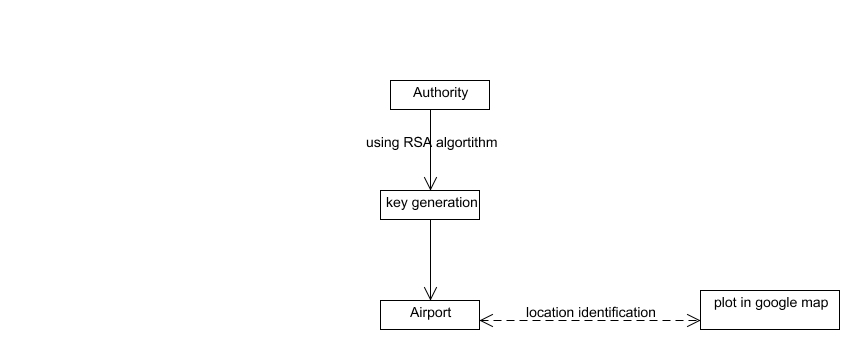


**4.5 Collaboration Diagram:**

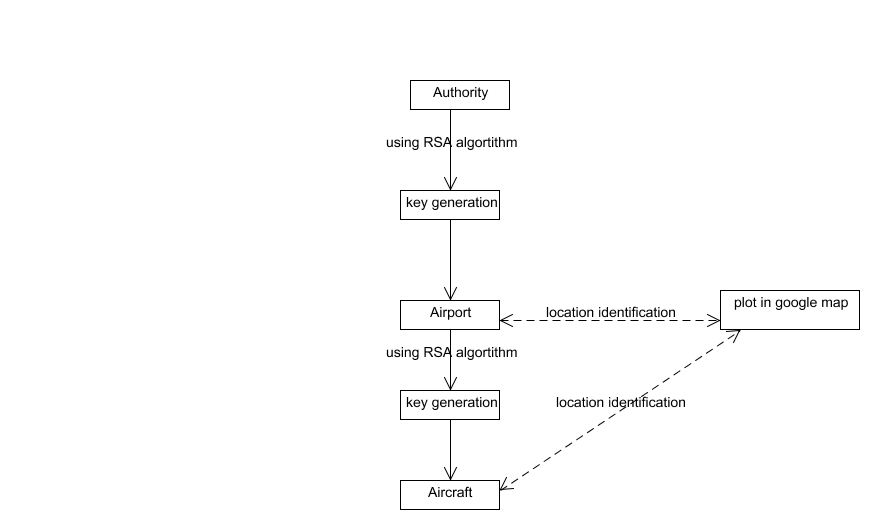
****

**4.6 Dataflow Diagram**

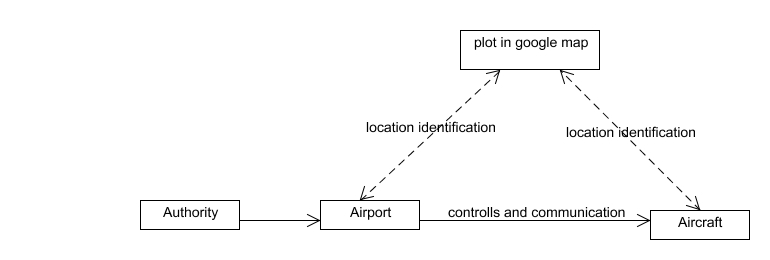
**Level 0**

****

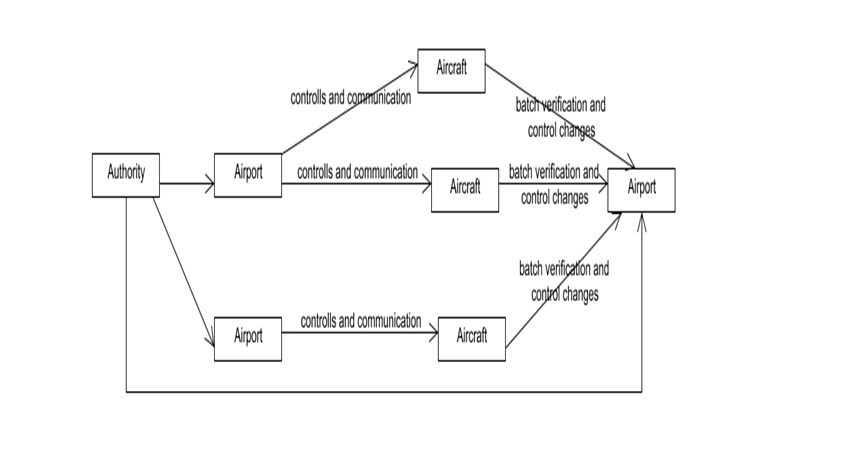
**Level 1**



**Level 2**

****

**Level 3**

****

**4.7 Class Diagram**

****

**5. SYSTEM DESIGN**

**5.1 MODULES**

* Web technology and smart objects,
* Virtual airport nodes and air craft nodes,
* Effective controlling and communications between Airports and crafts,
* Batch verification and control changes and identification of injected data.

**Keywords:**

APKG: Airline PKG

ASK: Airline Secret Key

PSK: Plane Secret Key

SKIDA: Airlines secret key

SKIDF: Aircraft secret key

**Web technology and smart object**:

Creating website to visually predict and plot our registered airports and aircrafts. The authorized elements from the root PKG (Airport authority of India) only can plot on Google map in our website. Using the highly secured networking socket communication can create the registered and authorized elements could appear on the map. The secured network application communicates through the authorized internet protocols among them. The root level main authority monitors the every occurrence of elements in the server of the website and verify with the Setup phase algorithm when the element get generated. The authorized element only gets the signatures and keys generated from the authority of root level PKG.

On input of a security parameter, the Setup algorithm allows the top PKG to generate the master secret key and the master public key. The root level PKG keeps the master key pair (msk, mpk) that will be used to generate secret keys for low-level nodes.

**Virtual airport and aircraft nodes:**

On input of the master secret key, and an airline identity IDA, the Extract A algorithm generates a secret key skIDA for IDA. The Level-1 nodes are airlines each with an identity like IDA. The second level PKGs which can generate the secret signing keys for the bottom level nodes.

Airport nodes are created to controls and communicate with the aircraft nodes. And craft nodes show the important details like range, location, destination, craft number and the source airport of the crafts themselves. Generation of the keys and ids are based on the input of top level PKG (Authority) to the low level PKG (Airlines and aircrafts) using RSA algorithm and HMAC algorithm. Due to this attackers cannot predict the keys and signatures of the nodes.

**Effective communications and controls between Airport and Aircraft:**

On input of the secret key skIDA for the Level-2 nodes are the aircraft each with an identity like IDF .The airline identity IDA and an aircraft identity IDF, the Extract F algorithm generates a secret key skIDF for IDF belonging to IDA.

The messages and information’s are high sensible and secured. So it could be highly integrated and secured. On the input of skIDF to the registered crafts are controlled and communicated by the source airports among the matched keys and signatures. And matched airports only can communicate with the aircrafts and controlled them automatically from belonged station. Information’s from the airport to aircrafts and crafts to station communications can be controlled by ground station belongs to the aircrafts. After reached the destination, above entire rolls are handled by the destination port.

**Batch verification and Control change and Identification of injected data:**

On input of the secret key skIDF and a message m, the Sign algorithm generates a signature of m. On input of a signature on a messages m with respect to airline identity IDA and aircraft identity IDF, the Verify algorithm outputs either 0 or 1, where 1 represents that the signature is valid.

In existing system each crafts are verified individually but proposed systems implements the batch verification schemes to verify the bulky requests from the crafts. We give a formal security proof for the proposed extended scheme that supports full batch verification. Verifications are based on the skIDF, skIDA, signature and keys from root level PKG’s. After the verification in destination port the entire controls are handle by the destination port which is alternately changes afterwards the verification in somewhere airports.

**CHAPTER 6**

**6.2 CODING STANDARDS**

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

* Program should be simple, clear and easy to understand.
* Naming conventions
* Value conventions
* Script and comment procedure
* Message box format
* Exception and error handling

**6.2.1 NAMING CONVENTIONS**

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

**Class names**

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

**Member Function and Data Member name**

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

**6.2.2 VALUE CONVENTIONS**

Value conventions ensure values for variable at any point of time. This involves the following:

* Proper default values for the variables.
* Proper validation of values in the field.
* Proper documentation of flag values.

**6.2.3 SCRIPT WRITING AND COMMENTING STANDARD**

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

**6.2.4 MESSAGE BOX FORMAT**

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

* X – User has performed illegal operation.
* ! – Information to the user.

**6.3 TEST PROCEDURE**

SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

* Static analysis is used to investigate the structural properties of the Source code.
* Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

**6.4 TEST DATA AND OUTPUT**

**6.4.1 UNIT TESTING**

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

**6.4.2 FUNCTIONAL TEST**

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

* Performance Test
* Stress Test
* Structure Test

**6.4.3 PERFORMANCE TEST**

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

**6.4.4 STRESS TEST**

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

**6.4.5 STRUCTURED TEST**

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

* Exercise all logical decisions on their true or false sides.
* Execute all loops at their boundaries and within their operational bounds.
* Exercise internal data structures to assure their validity.
* Checking attributes for their correctness.
* Handling end of file condition, I/O errors, buffer problems and textual errors in output information

**6.4.6 INTEGRATION TESTING**

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

**6.4.7 TESTING TECHNIQUES / TESTING STRATERGIES**

**a)TESTING**

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

**b) WHITE BOX TESTING**

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

* Flow graph notation
* Cyclometric complexity
* Deriving test cases
* Graph matrices Control

**c) BLACK BOX TESTING**

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

* Graph based testing methods
* Equivalence partitioning
* Boundary value analysis
* Comparison testing

**d) SOFTWARE TESTING STRATEGIES:**

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

* Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
* Different testing techniques are appropriate at different points in time.
* The developer of the software and an independent test group conducts testing.
* Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

**e) INTEGRATION TESTING:**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

**f) PROGRAM TESTING:**

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

**g) SECURITY TESTING**

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

**h) VALIDATION TESTING**

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

* The function or performance characteristics confirm to specifications and are accepted.
* A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic.

**i) USER ACCEPTANCE TESTING**

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

* Input screen design.
* Output screen design.

**Source Code**

**ADS-B.java**

package logic;

import java.io.ObjectInputStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.HashMap;

import java.util.StringTokenizer;

import java.util.Vector;

import org.omg.CORBA.PUBLIC\_MEMBER;

public class ADSBREC extends Thread {

public static Vector srcportvec=new Vector();

HashMap srcportmap=new HashMap();

HashMap destim=new HashMap();

int pnum=5678;

public static HashMap crafthm=new HashMap();

public ADSBREC()

{

//constructor used to run socket

start();

}

public ADSBREC(String s)

{

//constructor used to communicate cont1 and use vector

}

public void run()

{

try

{

ServerSocket sss=new ServerSocket(pnum);

while(true)

{

System.out.println(" \*\*\*\*\*\*\*\*\*\*ADSBATCH on successfully\*\*\*\*\*\* "+pnum);

Socket s=sss.accept();

System.out.println("------------ADSBATCH receiver connected-----------");

ObjectInputStream oip=new ObjectInputStream(s.getInputStream());

String str=oip.readObject().toString();

//System.out.println("------------"+str+"-----------");

StringTokenizer strtok=new StringTokenizer(str, "$");

String strtok1="",strtok2="",strtok3="",strtok4="",strtok5="",strtok6="";

strtok1=strtok.nextToken();

//System.out.println(strtok1);

if(strtok1.equalsIgnoreCase("location"))

{

strtok2=strtok.nextToken();

strtok3=strtok.nextToken();

strtok4=strtok.nextToken();

// strtok2=sourceAirport strtok3=portnumber

// System.out.println("In StartReceiver sourceAirport and portnumber-----"+strtok2+" "+strtok3);

srcportmap.put(strtok3, strtok2) ;

srcportvec.addElement(strtok2+"$"+strtok3+"$"+strtok4);

}

if(strtok1.equalsIgnoreCase("craft"))

{

strtok2=strtok.nextToken();

strtok3=strtok.nextToken();

strtok4=strtok.nextToken();

strtok5=strtok.nextToken();

strtok6=strtok.nextToken();

//strtok 2,3,4,5,6=craftnumber,source,destination, current craft latitude, longitude

crafthm.put(strtok2+"$"+strtok3+"$"+strtok4 , strtok5+"$"+strtok6 );

//System.out.println("In StartReceiver sourceAirport and portnumber-----"+strtok2+" "+strtok3);

}

//if()

}

}

catch(Exception e){

e.printStackTrace();

}

}

}

**Servlet.jsp**

package logic;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class servlet1 extends HttpServlet {

/\*\*

\* The doGet method of the servlet. <br>

\*

\* This method is called when a form has its tag value method equals to get.

\*

\* @param request the request send by the client to the server

\* @param response the response send by the server to the client

\* @throws ServletException if an error occurred

\* @throws IOException if an error occurred

\*/

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out

.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">");

out.println("<HTML>");

out.println(" <HEAD><TITLE>A Servlet</TITLE></HEAD>");

out.println(" <BODY>");

out.print(" This is ");

out.print(this.getClass());

out.println(", using the GET method");

out.println(" </BODY>");

out.println("</HTML>");

out.flush();

out.close();

}

**Cont1.jsp**

package logic;

import java.io.IOException;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.util.Iterator;

import java.util.Map;

import java.util.Set;

import java.util.StringTokenizer;

import java.util.Map.Entry;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletContextEvent;

import javax.servlet.ServletContextListener;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import com.sun.org.apache.bcel.internal.generic.NEW;

public class cont1 extends HttpServlet {

String time;

String desti;

ADSBREC recobj=new ADSBREC("call");

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

HttpSession ses = request.getSession(true);

String val1 = "", val2;

//System.out.println("cont1 call1");

val2=ADSBREC.srcportvec.lastElement().toString();

if(!ADSBREC.srcportvec.isEmpty())

{

StringTokenizer stringtoken1 = new StringTokenizer(val2, "$");

String srcloc = stringtoken1.nextToken();

String port = stringtoken1.nextToken();

String ip= stringtoken1.nextToken();

//ses.setAttribute("portnum", port);

out.print(srcloc+"$"+port+"$"+ip);

}

ADSBREC.srcportvec.clear();

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

HttpSession ses = request.getSession(true);

String latilongi = request.getParameter("latlng");

String source = request.getParameter("source");

String pnm = request.getParameter("portnum");

String ip=request.getParameter("ip");

System.out.println("in cont1=----"+ip);

//System.out.println("in sender--------" + pnm+"---"+latilongi.charAt(0));

if(! latilongi.equals(null) & ! latilongi.equals(""))

{

Senderjav sen = new Senderjav(latilongi+"$"+source,pnm,ip);

}

}

}

**Server.jsp**

package logic;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import com.sun.corba.se.spi.protocol.RequestDispatcherDefault;

public class sender extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

HttpSession ses = request.getSession(true);

String location = request.getParameter("latlng");

String pnm = ses.getAttribute("portnum").toString();

System.out.println("in sender--------" + pnm+"---"+location.charAt(0)+"-----"+location);

if(String.valueOf(location.charAt(0)).equals("("))

{

//Senderjav sen = new Senderjav(location,pnm);

}

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out

.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">");

out.println("<HTML>");

out.println(" <HEAD><TITLE>A Servlet</TITLE></HEAD>");

out.println(" <BODY>");

out.print(" This is ");

out.print(this.getClass());

out.println(", using the POST method");

out.println(" </BODY>");

out.println("</HTML>");

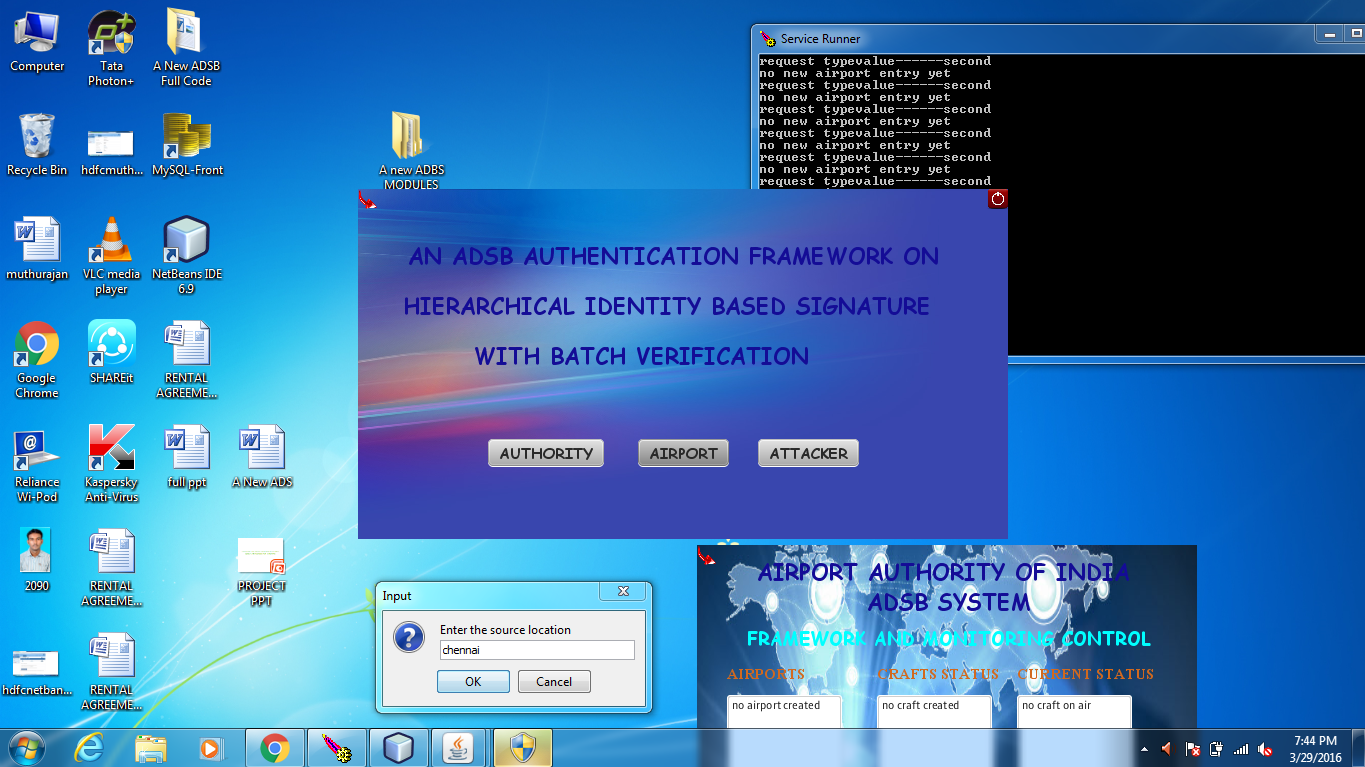
out.flush();

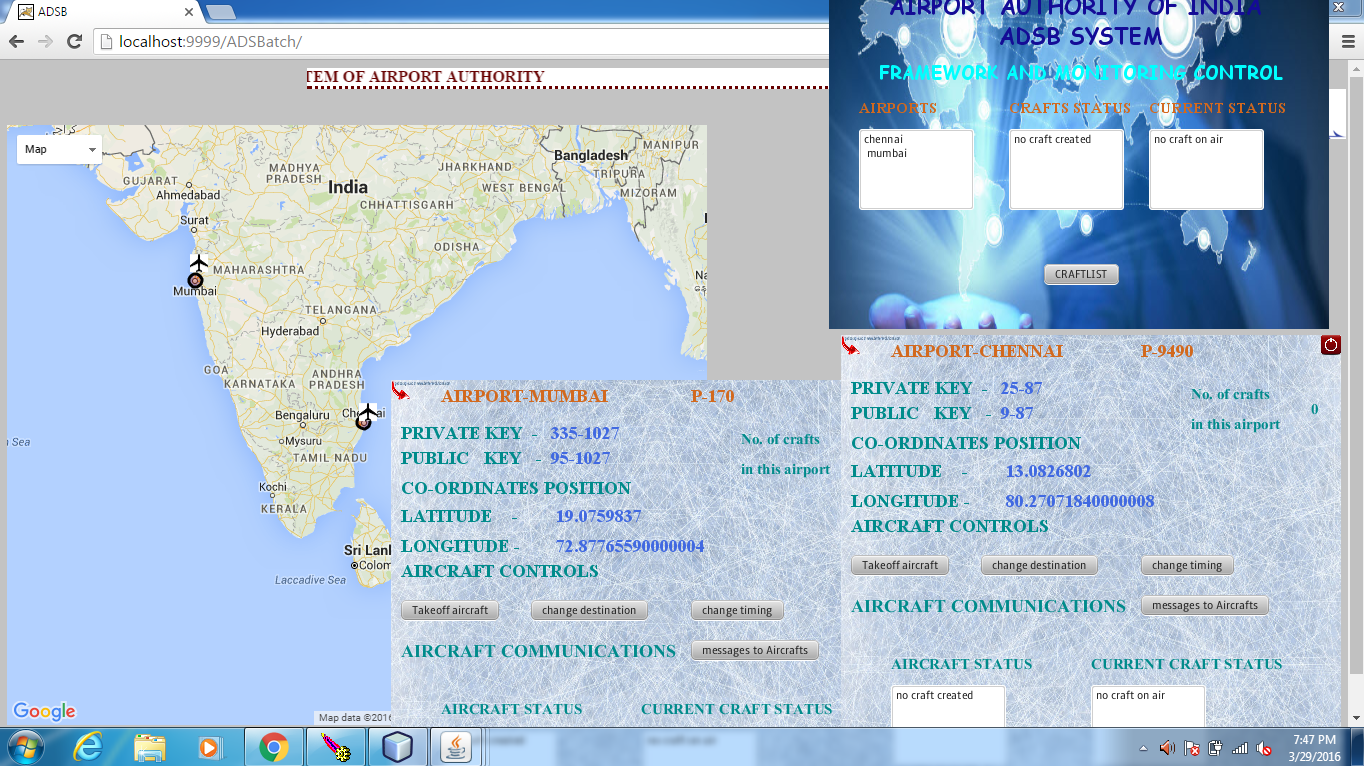
out.close();

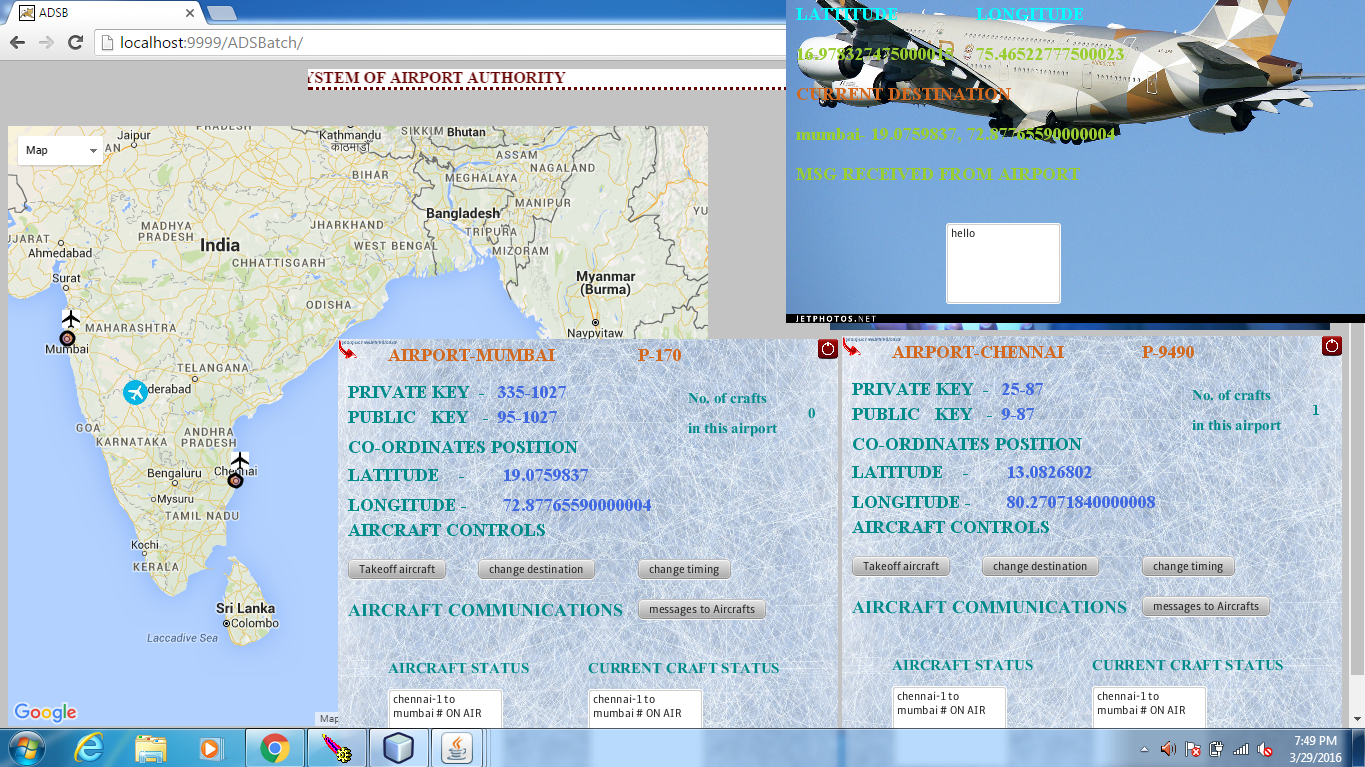
}

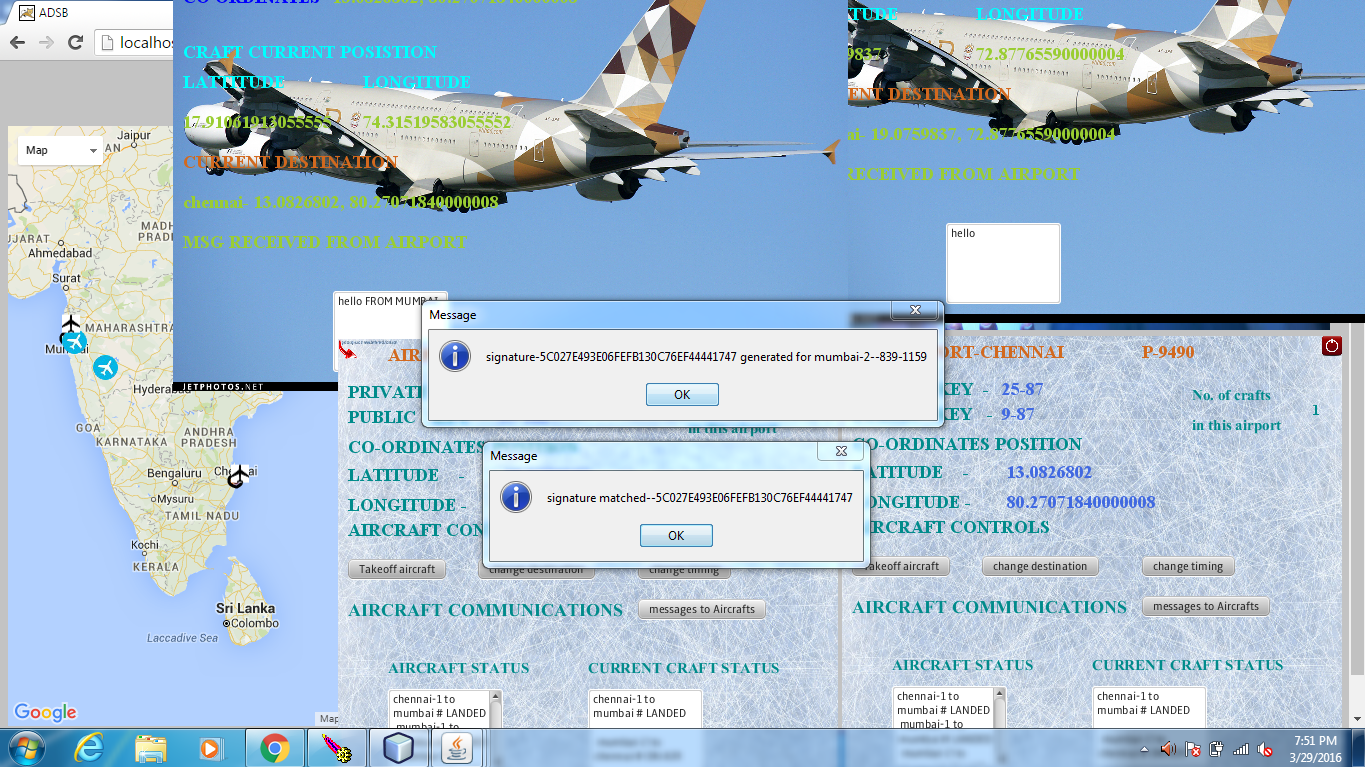
}

**Screenshots**











**REFERENCES**

[1] http://www.flightradar24.com

[2] Baek, J., Byon, Y.J., Hableel, E., Al-Qutayri, M.: An authentication framework for automatic dependent surveillancebroadcast based on online/offline identity-based signature. In: 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2013. pp. 358–363. IEEE (2013)

[3] Baek, J., Byon, Y.J., Hableel, E., Al-Qutayri, M.: Making air traffic surveillance more reliable: a new authentication framework for automatic dependent surveillance-broadcast

(ads-b) based on online/offline identity-based signature Security and Communication Networks (2014)

[4] Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Advances in Cryptology - EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–

250. Springer (1998)

[5] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. Journal of Cryptology 17(4), 297–319 (2004)

[6] Camenisch, J., Hohenberger, S., Pedersen, M.Ø.: Batch verification of short signatures. In: Advances in Cryptology - EUROCRYPT 2007. LNCS, vol. 4515, pp. 246–263. Springer

(2007)

[7] Costin, A., Francillon, A.: Ghost in the air (traffic): O insecurity of ADS-B protocol and practical attacks on ADS devices. Black Hat USA (2012)

[8] Feng, Z., Pan, W., Wang, Y.: A data authentication solution of ADS-B system based on x.509 certificate. In: 27th Internationa Congress of the Aeronautical Sciences, ICAS 2010 (2010)

[9] Ferrara, A.L., Green, M., Hohenberger, S., Pedersen, M.Ø.: Practical short signature batch verification. In: Topics in Cryptology - CT-RSA 2009. LNCS, vol. 5473, pp. 309–324. Springer (2009)

[10] Fiat, A.: Batch rsa. In: Advances in Cryptology - CRYPTO’89. LNCS, vol. 435, pp. 175–185. Springer (1989)