



**SECURE CONTENT PROCESSING AND DISTRIBUTION BY  
INTERMEDIARIES**

**By**

**UMA MALINI.K**

**Reg. No: 0820108022**

**of**

**KUMARAGURU COLLEGE OF TECHNOLOGY**  
**(An Autonomous Institution Affiliated to Anna university, Coimbatore )**

**COIMBATORE – 641 006**

**A PROJECT REPORT**

**Submitted to the**

**FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING**

*In partial fulfillment of the requirements  
for the award of the degree  
of*

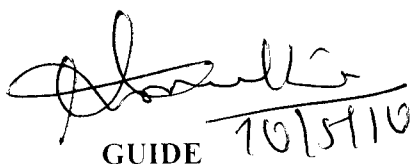
**MASTER OF ENGINEERING  
IN**

**COMPUTER SCIENCE AND ENGINEERING**

**MAY, 2010**

## BONAFIDE CERTIFICATE

Certified that this project report titled “**Secure Content Processing and Distribution by Intermediaries**” is the bonafide work of **UMAMALINI.K (0820108022)** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report of dissertation on the basis of which a degree or ward was conferred on an earlier occasion on this or any other candidate.



GUIDE 18/5/10

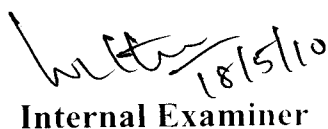
(Mrs. AMUTHA VENKATESSH)



HEAD OF THE DEPARTMENT

(P.DEVAKI)

The candidate with **University Register No. 0820108022** was examined by us in Project Viva-Voce examination held on 18/5/10



Internal Examiner



External Examiner



# VIVEKANANDHA

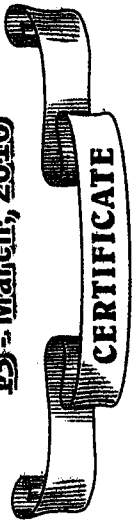
INSTITUTE OF ENGINEERING AND TECHNOLOGY FOR WOMEN  
ELAYAMPALAYM, TIRUCHENGODE.



Department of Computer Applications

## NATIONAL CONFERENCE ON " EMERGING TECHNOLOGIES IN ADVANCED COMPUTING AND COMMUNICATION "

13<sup>th</sup> - March, 2010



This is to Certify that Mr. / Ms. / Dr. K.UMA MALINI / J. ME. (CSE.).....

KUHARAGURU COLLEGE OF TECHNOLOGY..... has participated in t

" National Conference on ETACC '10 ", organized by " VIVACIOUS " the professional association

Computer Applications on 13th March 2010 and presented a paper on SECURE CONTENT PROCESS

AND DISTRIBUTION BY COOPERATIVE INTERMEDIARIES.....

*A. Thomas kothal*

HOD & Organizing Secretary

*V. h...*

Principal

*[Signature]*

Chairman & Secret

# ABSTRACT

Content services such as content filtering and trans-coding adapt contents to meet system requirements, display capacities, or user preferences. Data security in such a framework is an important problem and crucial for many Web applications. This approach addresses data integrity and confidentiality in content adaptation and caching by intermediaries.

This approach also permits multiple intermediaries to simultaneously perform content services on different portions of the data. The protocol supports decentralized proxy, key management and flexible delegation of services.

It is a solution for secure content services characterized by a scalable and robust network architecture. The protocol allows a client to verify that the received data is authentic and transformations on the data are properly authorized. This approach also assures data confidentiality during transmission. It also highlights load distribution through proxies to improve system performance and supports parallel content services.

## ஆய்வுச்சுருக்கம்

ஒரு பிணையத்தில் அங்கிகாரம் பெற்ற கணினிகள் பல உள்ளன அதில் அங்கிகாரம் பெறாத உறுப்பினர்கள் சில தேவையற்ற தரவுகளை வழங்கிக்கு அனுப்புகொண்டே இருக்கும், அப்படி அனுப்பும் போது வழங்கி செயலிழந்து அங்கிகாரம் பெற்ற உறுப்பினர்களுக்கு தேவையான செய்திகளை தர முடியாமல் போய்விடுகிறது அது ஒரு பிணையத்தின் செயல்பாட்டை குறைக்கும்.

அந்த தேவையற்ற தரவுகளால் ஒரு பிணையத்தின் வேகமான செயல்பாட்டை மிகவும் குறைக்கும் அதனால் பணி இடங்களில் ஒழுங்கான பிணையத்தை உருவாக்க முடியாமல் போகிறது.

இந்த ஆய்வில் ஏசிஎஸ் வழங்கி என்ற உறுப்பை பயன்படுத்தி உள்வாங்கும் தரவுகளை சோதித்து தேவையற்ற தரவுகளை நீக்கி விட்டு தேவையான தரவை மட்டும் வழங்கிக்கு அனுப்பும்.

மேலும் பிறரிடம் இருந்து தகவல்களை மேலும் பாதுக்காக்க ஏ இ எஸ் எங்கின்ற தகவல் உருமாற்றும் தந்திரம் உபயோகப்படுத்தி தகவல்கள் உறுமாற்றி பாதைகளையும் கட்டமைப்புகளையும் தேர்வு செய்யும் திறனானது பெரும்பாலான தகவல் திருடல் மற்றும் தடுத்தல்களில் இருந்து தகவல்களை காக்கின்றது..

மேற்கூறியவற்றால் தகவல் பறிமாற்றமான மிகவும் நம்பகத்தன்மையாக உள்ளது. அனுப்புனர் மற்றும் பெறுநர் மட்டுமே தகவல்களை அறியும்படியான பாதுக்காப்பாக அனுப்ப ஏதுவாக உள்ளது.

## ACKNOWLEDGEMENT

I express my profound gratitude to our Chairman **Padmabhusan Arutselvar Dr.N.Mahalingam B.sc, F.I.E** and Correspondent **Shri.Balasubramanian M.Com, M.B.A (USA)** and joint Correspondent **Dr.A.Selvakumar Ph.D.**, for giving this great opportunity to pursue this course.

I thank **Dr.S.RamaChandran**, Principal, Kumaraguru College of Technology, Coimbatore, for providing me with the necessary facilities and Infrastructure to work on this project.

I express my sincere thanks to **Dr.S.Thangasamy**, Professor and Dean, Department of Computer Science and Engineering, for being my greatest of inspiration and for embedding the quest for innovative ideas.

I express my thanks to Mrs. P.Devaki, H.O.D Incharge, CSE Dept who has helped in doing this project.

I record my thanks to the My Guide **Mrs. Amutha Venkatesh, M.E.**, Assistant Professor, Department of Computer Science and Engineering, who has extraordinarily helped in preparing this project work successfully.

I express my deep sense of gratitude and gratefulness to Project Coordinator **Mrs.V.Vanitha, M.E, (Ph.D)**, Asst. Professor, Department of Computer Science and Engineering, for her supervision, tremendous patience, active involvement and guidance.

I would like to convey my honest thanks to all Teaching staff members and Non Teaching staff members of the department for their support. I would like to thank all my classmates who gave me proper light moments and study breaks apart from extending some technical support whenever I needed them most.

I dedicate this Project work to my **Parents** for no reasons but feeling from bottom of my heart, without their love this work wouldn't possible.

# TABLE OF CONTENTS

<b>CONTENTS</b>	<b>PAGE NO</b>
<b>Abstract</b>	<b>iii</b>
<b>Abstract (Tamil)</b>	<b>iv</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Abbreviations</b>	<b>ix</b>
<b>1. INTRODUCTION</b>	
1.1 Overview Of Network Security	1
1.2 Networks	1
1.3 Content distribution network	2
1.4 Security issues	3
1.5 Symmetric key cryptography	4
1.6 Digital signature	4
1.7 Key Distribution	4
<b>2. LITERATURE REVIEW</b>	
2.1 Data Integrity service model	5
2.2 Content adaptation	6
2.3 XML update protocol	8
2.4 Advanced Encryption Standard (AES) Algorithm	9
2.4.1 Description Of The Cipher	9
2.4.2 High-Level Description Of The Algorithm	9
<b>3. METHODOLOGY</b>	
3.1 Overview	11
3.2 Understanding Existing Approaches	11

3.3 Proposing System	12
3.3.1 Problem definition	12
3.3.2 System model	12
3.3.3 System architecture	13
3.3.4 Modules	14
3.3.4.1 Client	14
3.3.4.2 Access Control System	15
3.3.4.3 Intermediaries	17
3.3.4.4 Data Server	18
<b>4. TECHNOLOGY INFRASTRUCTURE</b>	
4.1 Core Java	20
4.2 Swing	20
4.3 ODBC	22
4.4 JDBC	25
4.5 Networks	27
<b>5. EXPERIMENTAL RESULTS</b>	
5.1 Client authentication	30
5.2 Client registration	30
5.3 Client view	32
5.4 File downloading	33
5.5 Play video	34
5.6 Decompression	34
5.7 Data server	35
5.8 Database	36
<b>6. CONCLUSION AND FUTURE OUTLOOK</b>	<b>37</b>
<b>APPENDIX</b>	<b>38</b>
<b>REFERENCES</b>	<b>46</b>



# LIST OF FIGURES

FIGURE NO	CAPTION	PAGE NO
3.1	System Architecture	13
3.2	Client process	15
3.3	Access control system	17
3.4	Intermediary	18
3.5	Data server	19
3.6	Swing class	23
3.7	TCP/IP Stack	27
3.8	IP Addressing	28

## LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
WLANS	Wireless Local Area Networks
DISM	Data Integrity service Model
DP	Data Provider
DES	Data Encryption Standard
AES	Advanced encryption Standard
GUI	Graphic User Interface
AWT	Abstract Window Tool
JFC	Java Foundation Classes

# INTRODUCTION

## 1.1 OVERVIEW OF NETWORK SECURITY

In the field of networking, the specialist area of network security consists of the provisions made in an underlying computer network infrastructure, policies adopted by the network administrator to protect the network and the network-accessible resources from unauthorized access, and consistent and continuous monitoring and measurement of its effectiveness (or lack) combined together.

Network security involves all activities that organizations, enterprises, and institutions undertake to protect the value and ongoing usability of assets and the integrity and continuity of operations.

## 1.2 NETWORKS

There are wired and wireless networks. In the recent years wireless networks have witnessed a tremendous increase of popularity in both research and industry. There are currently two variations of mobile networks. The first is widely known as infrastructure networks since the gateways that connect them to other networks (like the internet) are fixed and wired. The bridges in these networks are also known as base stations. In an environment like this, a node is able to roam freely and establish a connection link with the nearest base station that is within its communication range. As the mobile node moves out of the base station that it was connected with, it falls into the range of another and hand off occurs between the old base station and the current one, enabling the mobile unit to continue communication seamlessly through the network. These types of networks are most widely applied in office areas.

The second type of wireless networks is the infrastructure less mobile network that is also known as an ad hoc network.

### 1.3. CONTENT DISTRIBUTION NETWORK

In order to enhance the performance of content distribution networks (CDNs), several approaches have been developed based on the use of content management services provided by intermediary proxies. In most of these approaches, content caching is the main service provided by proxies. That is, instead of asking a content server for contents upon each client request, a proxy first checks if these contents are locally cached. Only when the requested contents are not cached or out of date are the contents transferred from the content server to the clients. If there is a cache hit, the network bandwidth consumption can be reduced. A cache hit also reduces access latency for the clients. System performance thus improves, especially when a large amount of data is involved. Besides these improvements, caching makes the system robust by letting caching proxies provide content distribution services when the server is not available. With the emergence of various network appliances and heterogeneous client environments, there are other relevant new requirements for content services by intermediaries.

Many studies have been carried out on intermediary content services; however, the problem of data security in these settings has not caught much attention. Confidentiality and integrity are two main security properties that must be ensured for data in several distributed cooperative application domains such as collaborative, distance learning, telemedicine, and e-government. Confidentiality means that data can only be accessed under the proper authorizations. Integrity means that data can only be modified by authorized subjects. The approaches developed for securely transferring data from a server to clients are not suitable when data is to be transformed by intermediaries.

When a proxy mediates data transmission, if the data is enciphered during transmission, security is ensured; however, it is impossible for intermediaries to modify the data. On the other hand, when intermediaries are allowed to modify the data, it is difficult to enforce security.

1. **Data Server:** This is an entity that originally stores the data requested by a client.
2. **Client:** This is any entity that requests data from a data server. When a client submits a request, besides the data it requests, it may also include some content service requirements, arising from device limitations and data format limitations. If the client does not specify any service requirements, a proxy that represents the client may add these requirements. Such a proxy may be an edge proxy.
3. **Intermediary:** This is any entity that is allowed by a data server to provide content services in response to requests by clients. Intermediaries include caching proxies and transforming proxies.

## 1.4. SECURITY ISSUES

There are two main security properties that must be ensured for data in several distributed cooperative applications

- Confidentiality
- Integrity

Confidentiality means that data can only be accessed under the proper authorizations. Integrity means that data can only be modified by authorized subjects.

## 1.5. SYMMETRIC KEY CRYPTOGRAPHY

It is a class of algorithms for cryptography that use trivially related, often identical, cryptographic keys for both decryption and encryption. The encryption key is trivially related to the decryption key, in that they may be identical or there is a simple transformation to go between the two keys. The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link. Other terms for symmetric-key encryption are secret-key, single-key, shared-key, one-key, and private-key encryption. Use of the last and first terms can create ambiguity with similar terminology used in public-key cryptography.

A digital signature or digital signature scheme is a mathematical scheme for demonstrating the authenticity of a digital message or document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, and that it was not altered in transit. Digital signatures are commonly used for software distribution, financial transactions, and in other cases where it is important to detect forgery and tampering.

## **1.7. KEY DISTRIBUTION**

In symmetric key cryptography, both parties must possess a secret key which they must exchange prior to using any encryption. Distribution of secret keys has been problematic until recently, because it involved face-to-face meeting, use of a trusted courier, or sending the key through an existing encryption channel. The first two are often impractical and always unsafe, while the third depends on the security of a previous key exchange.

In public key cryptography, the key distribution of public keys is done through public key servers. When a person creates a key-pair, he keeps one key private and the other, public-key, is uploaded to a server where it can be accessed by anyone to send the user a private, encrypted, message.

## LITERATURE SURVEY

### 2.1 DATA INTEGRITY SERVICE MODEL:

It is an XML-based solution to address data integrity. DISM is an in-markup XML based language designed to preserve the integrity of the web pages. DISM is compatible with HTTP protocol. It accepts normal HTTP requests; and the DISM response message can pass through non-DISM proxy without causing any error.

DISM allows the content owner to specify modification policies on the web contents. These policies indicate when, how and by whom the content can be changed. The policies can be attached to the server's response message and delivered together with the content. DISM also allows these policies to be delegated to other trusted parties.

DISM provides an authorization and validation model to protect the response message from being tampered. This security model is based on the public-key cryptosystem. It allows any party to validate the message with respect to the server's digital signature. The W3C XML signature standard serves as a foundation for the definition of this model.

DISM allows the content owner to divide the web content into separate parts and assign each part different cacheability profile. It also allows proxies to treat parts of web objects as cacheable resources. This gives the client the flexibility to request for partial cache invalidation, therefore, reducing both the required bandwidth and the server workload.

In DISM, the basic unit for authorization, modification and validation is "segment". A web page is divided into a set of segments each of which possesses its own content, modification rights and modification record. The collection of these segments is a "manifest". The modification rights of the segments are called "permissions". Permissions specify how their parent segment can be modified. A complete permission contains the information about type of the permitted "action"(E.g. add, delete, replace), conditions under which the permission can be executed, identity of the authorized "editor" and the authorizer's digital signature. Some permission can be further delegated to other trusted party if the "delegatable" attribute is set to be "yes". Nevertheless, issuing new permission is beyond the right of the proxies. The security policy of DISM is "monotonic" in the sense that the number available

In order for the client to validate the received message, each editor must leave a modification record in the corresponding segment. These modification records indicate when, how and by whom the content was modified. The modification records are secured by the editor's signatures. DISM is a cache-friendly model in the sense that the segments are relatively independent of each other and they can be separately cached and invalidated. DISM to enforce the integrity of data transformed by intermediaries. In such a model, integrity is enforced by using metadata expressing modification policies specified by content owners.

Limitations:

User adaptation is an important aspect but we require an individual proxy for each adaptation. This process of providing individual proxies to each adaptation makes the process slow.

## **2.2. CONTENT ADAPTATION**

In today's Internet, there is a wide range of client devices in terms of both hardware and software capabilities. Device capabilities vary in different dimensions, including processing power, storage space, display resolution and color depth, media type handling, and much more. This variety on device capabilities makes it extremely difficult for the content providers to produce a content that is acceptable and appreciated by all the client devices, making application-level adaptation a necessity to cover the wide variety of clients.

There are three main approaches for handling this diversity in content formats: a static content adaptation, a dynamic content adaptation, and a hybrid of the static and dynamic approaches.

The first two approaches differ in the time when the different content variants are created to match the requested format. In static adaptation, the content creator generates and stores different variants of the same content on a content server, with each variant formatted for a certain device or class of devices.

Static adaptation has three main advantages: It is highly customized to specific classes of client devices, and it does not require any runtime processing, so no delay is incurred, and the content creator has the full control on how the content is formatted and delivered to the client. On the other hand, static adaptation has a number of disadvantages, mainly related to the management and maintenance of different variants of the same content : different content



the same content.

In the server-based approach, the content server is responsible for performing the trans-coding; the content provider has all the control on how the content is transcoded and presented to the user. Additionally, it allows the content to be trans-coded before it is encrypted, making it secure against malicious attacks. On the other hand, server-based adaptation does not scale properly for a large number of users and requires high-end content and delivery server to handle all requests.

As for the client-based approach, the client does the trans-coding when it receives the content. The advantage of this approach is that the content can be adapted to match exactly to the characteristics of the client. But at the same time, client-based adaptation can be highly expensive in terms of bandwidth and computation power, especially for small devices with small computational power and slow network connectivity, with large volume of data might be wastefully delivered to the device to be dropped during trans-coding.

The third adaptation approach is the proxy-based approach, where an intermediary computational entity can carry out content adaptation on the fly, on behalf of the server or client. Proxy adaptation has a number of benefits including leveraging the installed infrastructure and scaling properly with the number of clients. It also provides a clear separation between content creation and content adaptation.

### **2.3. XML UPDATE PROTOCOL**

The subjects participating in the updates are cooperative. The Completion of the update depends on each subject. If one subject cheats more than twice, a receiver will notify DS and DS may broadcast that the updates failed and aborted.

DS has access to the flow policies and to the security policies of the document. The DS is a trusted entity. It determines these policies before the update process starts. Then these policies are enforced and are not modified during the execution of the update process.

There is no collusion among the subjects. Each subject does not share information with other subjects. The server protocol includes the following steps: construct the S-RPF graph, generate and send each subject its own control information, and send to the first subject(s) the encrypted package(s).

contains the symmetric key for the receiver to decrypt an incoming package; it also includes the sequence of regions the incoming package will contain, and for each region the public key of the last subject who authored this region. The goal of an incoming package template is to help a receiver to verify that the package it receives is from a specified sender and to verify that the content of the package is correct up to that point. Different subjects will receive different incoming templates from DS. An outgoing package template includes the symmetric key for the sender to encrypt the package and the sequence of regions to be sent in this package, so the sender can organize a package for its successor with the correct content.

During document updates, each subject executes the following steps: (i) it performs integrity check according to incoming package templates received from DS; (ii) it executes operations on the document according to its privileges; (iii) it forms packages according to outgoing package templates received from DS, and sends out these packages.

## **2.4 ADVANCED ENCRYPTION STANDARD (AES) ALGORITHM**

In cryptography, the AES is an encryption standard adopted by the U.S. government. The standard comprises three block ciphers, AES-128, AES-192 and AES-256, adopted from a larger collection originally published as Rijndael. Each of these ciphers has a 128-bit block size, with key sizes of 128, 192 and 256 bits, respectively. The AES ciphers have been analyzed extensively and are now used worldwide, as was the case with its predecessor the Data Encryption Standard (DES).

### **2.4.1 Description Of The Cipher**

AES is based on a design principle known as a Substitution permutation network. It is fast in both software and hardware. Unlike its predecessor, DES, AES does not use a feistel network. AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits, whereas Rijndael can be specified with block and key sizes in any multiple of 32 bits, with a minimum of 128 bits. The block size has a maximum of 256 bits, but the key size has theoretically no maximum.

AES operates on a  $4 \times 4$  array of bytes, termed the *state* (versions of Rijndael with a larger block size have additional columns in the state). Most AES calculations are done in a

of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key.

## **2.4.2 High-Level Description Of The Algorithm**

### **2.4.2.1 Key Expansion**

Round keys are derived from the cipher key using Rijndael's key schedule.

### **2.4.2.2 Initial Round**

- **AddRoundKey** :- Each byte of the state is combined with the round key using bitwise xor.

### **2.4.2.3 Rounds**

- **SubBytes**:- A non-linear substitution step where each byte is replaced with another according to a lookup table.
- **ShiftRows**:- A transposition step where each row of the state is shifted cyclically a certain number of steps.
- **MixColumns**:- A mixing operation which operates on the columns of the state, combining the four bytes in each column
- **AddRoundKey**

### **2.4.2.4 Final Round**

- **SubBytes**
- **ShiftRows**
- **AddRoundKey**

# METHODOLOGY

## 3.1 OVERVIEW

The approach used can be classified in 2 steps: (i) understanding the problem and existing solutions, (ii) implementing and evaluating proposed solution.

## 3.2 EXISTING APPROACHES

Data Integrity Service Model(DISM) is an in-mark up language designed to preserve the integrity of the web pages. DISM is compatible with HTTP protocol. It accepts normal HTTP requests; and the DISM response message can pass through non-DISM proxy without causing any error.

DISM allows the content owner to specify modification policies on the web contents. These policies indicate when, how and by whom the content can be changed. The policies can be attached to the server's response message and delivered together with the content. DISM also allows these policies to be delegated to other trusted parties. DISM provides an authorization and validation model to protect the response message from being tampered. This security model is based on the public- key cryptosystem. It allows any party to validate the message with respect to the server's digital signature. The W3C XML signature standard serves as a foundation for the definition of this model.

DISM allows the content owner to divide the web content into separate parts and assign each part different cacheability profile. It also allows proxies to treat parts of web objects as cacheable resources. This gives the client the flexibility to request for partial cache invalidation, therefore, reducing both the required bandwidth and the server workload.

This method addresses Integrity. But still there is a lack of Confidentiality.

### 3.3.1 PROBLEM DEFINITION

It is a method to improve Integrity and confidentiality in providing content services and caching by intermediaries. when intermediaries perform content services, there is possibility for intruder to perform some illegal operations on the data. so it is a threat to confidentiality and integrity. When a proxy mediates data transmission, if the data is enciphered during transmission, security is ensured; however, it is impossible for intermediaries to modify the data. On the other hand, when intermediaries are allowed to modify the data, it is difficult to enforce security.

Confidentiality and integrity are two main security properties that must be ensured for data in several distributed cooperative applications domains such as collaborative e-commerce, distance learning, telemedicine, and e-government. Confidentiality means that data can only be accessed under the proper authorizations. Integrity means that data can only be modified by authorized subjects.

### 3.3.2 SYSTEM MODEL

#### 3.3.2.1 Data Server:

This is an entity that originally stores the data requested by a client.

#### 3.3.2.2 Client:

This is any entity that requests data from a data server. When a client submits a request, besides the data it requests, it may also include some content service requirements, arising from device limitations and data format limitations. If the client does not specify any service requirements, a proxy that represents the client may add these requirements. Such a proxy may be an edge proxy.

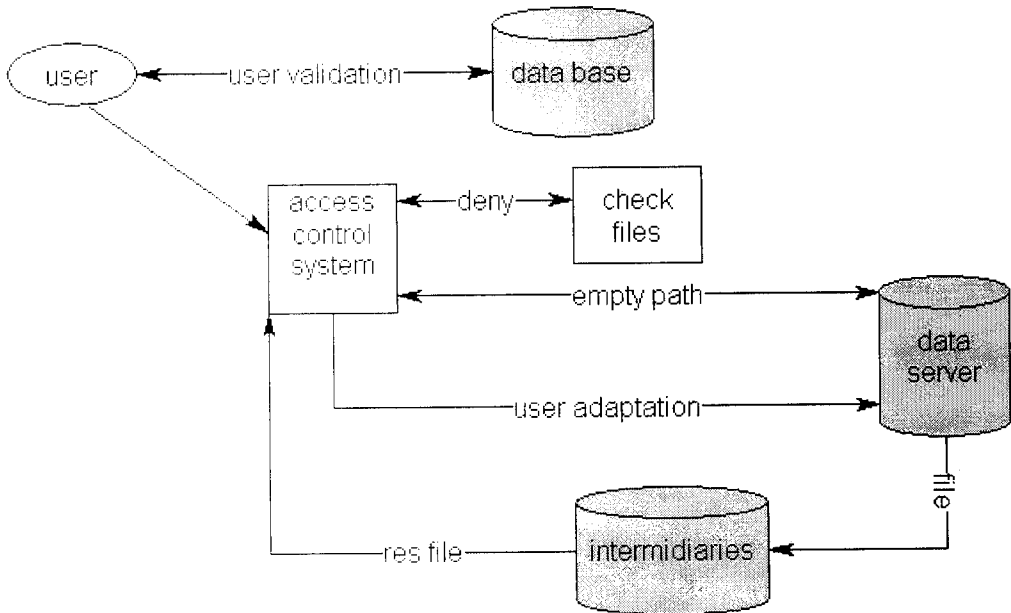
P-3296



response to requests by clients. Intermediaries include caching proxies and transforming proxies.

### 3.3.3 System Architecture:

The process of the design implemented with the system architecture view comprises of the parts of the project work that encapsulates all modules ranging from module to module communication, setting initializations and system.



**Fig.3.1. System Architecture**

The diagram shows the four modules and the operations performed in each module. In this diagram, we are showing all over the structure of the project. Here, the user first registers the information to database. Then the user authentication is checked by the database server. The access control system gets the request and process it. If it is normal request it will get the file content from data server.

the corresponding client.

### **3.3.4 MODULES**

There are four Modules. The module description is as follows,

#### **3.3.4.1 CLIENT**

This is an entity that requests data from a data server. When a client submits a request, besides the data it requests, it may also include some content service requirements, arising from device limitations and data format limitations .If the client does not specify any service requirements, a proxy that represents the client may add these requirements. Such a proxy may be an edge proxy.

In client request that client can select their options they need. According to the client request the request process will begin. Here, the user adaptation and file names are attach in the request header. That request header information will check on access control system that is in our second module.

In client system will give four trans-coding options for users.

The four trans-coding options are,

- Converting text file to image file
- Converting color image to gray image
- Compressing file
- Creating zip-file

The user adaptation will select based on the system requirement and user need.

Next the download process, user can verify the data integrity checking. Before that the ACS enforce the confidentiality in user file. The encrypted file also decrypt in client side. In our process we ensure the client authentication and confidentiality.The additional part of our process is, validating the form field in client page. Another one is registering new user in our application.

we just register the user information and check whether the user record is available or not in user record table.

Here the part of the work will be done in this process.



**Fig.3.2. client process**

### 3.3.4.2 ACCESS CONTROL SYSTEM

The three steps of ACS. They are discussed as follows,

- Access Deny
- Empty Path
- Path with information

Access Deny:

This indicates that the DP does not have the data requested by the client, the client is not allowed to access the data according to the DP's policy, or no intermediaries in the DP's intermediary profile table exist or are allowed to transform the data into the version requested by the client.

Empty Path:

This indicates that the client's request can be satisfied without any intermediary's involvement.



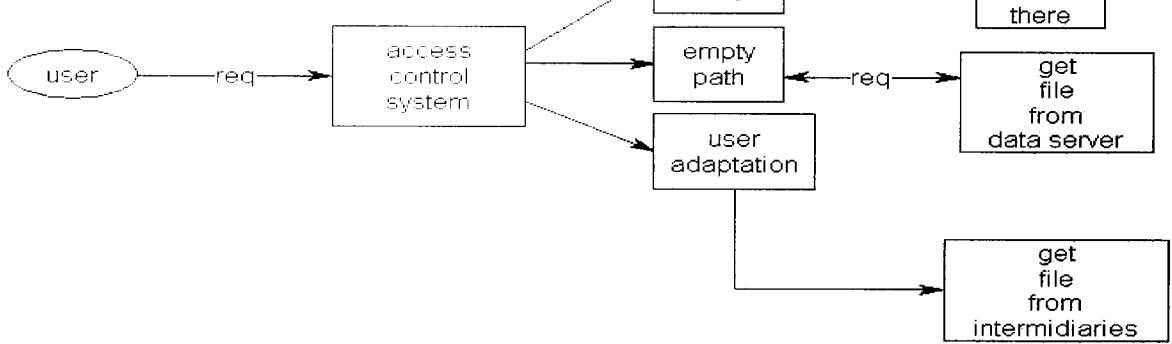
proxies listed in the returned path. ACIS denotes access control information structure, which specifies the privileges over the data for each Proxy in the path.

In ACS we receive the client request then we analysis the request information. In this request information first we analysis whether the request is empty-path or path with information.

In this empty-path we can get the file directly from data-server. There is no need for format changing in empty-path. Getting the file from data-server is another part of process in ACS. Here, we have to send the request to data-server and getting the file from data-server.

In ACS we just encrypt the file and send the file to client for enforcing the confidentiality. AES algorithm is used for encrypting the file. In this AES algorithm we used 128-bit key and 16 round for encryption.

Another part of request is path with information, in this we pass the request to data-server and get the content from the file and send the content to intermediaries based on the request. Access deny is our first step of this process, in this step we checking the file authorization that is whether the file can be modify or not. After that here we are checking the user authorization. According to the user authorization and authentication we send the file to the user.



**Fig.3.3. Access Control System**

### 3.3.4.3 INTERMEDIARIES

This is any entity that is allowed by a data server to provide content services in response to requests by clients. Intermediaries include caching proxies and transforming proxies.

In intermediaries we have proxies and cache memory. In this module we change the file format and send it to the ACS.

In this project we have two intermediaries and four proxies. Each intermediary has two proxies.

Intermediary one has two proxies the proxies are,

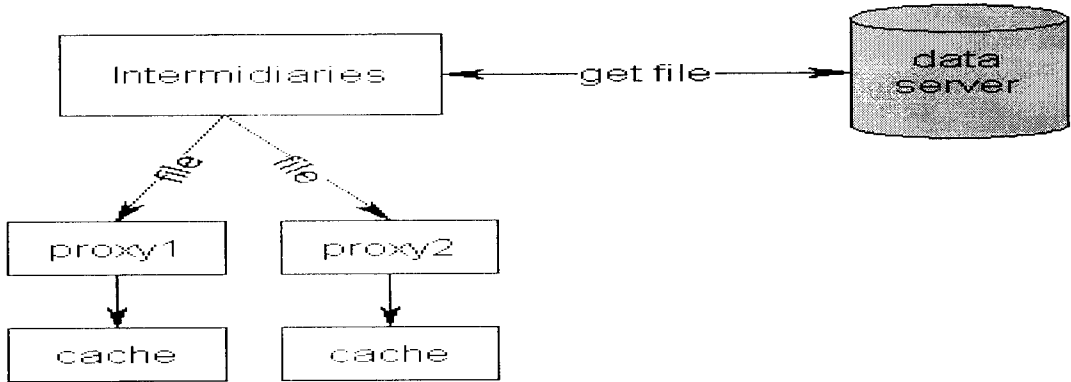
- Compressing the file
- Zip file

Intermediary two has two proxies the proxies are,

- Text to image file
- Color Image to Gray Image

One part of intermediary work is, receiving the packets from data server. In this part we are implementing the control information algorithm to receive and grouping the segments.

These intermediaries receive the file from data server in segment by segment and group all the segment in one file. Identifying the corresponding proxy is next process. In this process each proxy has one public key. Intermediary used this symmetric key to identifying the corresponding proxy. When the trans-code completed that intermediary take the trans-coded file into cache memory.



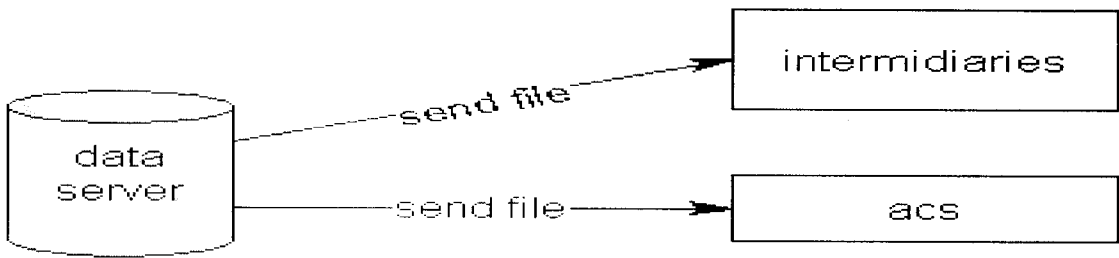
**Fig.3.4. Intermediary**

#### **3.3.4.4 DATA SERVER**

This is an entity that originally stores the data requested by a client. In this data-server we store all the files in this place. This is the server just give the response to the client. Before the response we have to check whether the user has authorization or not. If she/he has the authorization for this particular file the data-server then search the file and send it to ACS. Before send the file content to ACS. It will check the user adaptation. If it is empty path it just send the file content to ACS otherwise it will send the file to Intermediary.

It used for store the files and keep proxy table. Using proxy table it will find the appropriate intermediary for send that file.

The data server needs to send segments to corresponding P-proxies or to the client. Each P-proxy must receive the segments that are allowed to access and send some or all of these segments to subsequent P-proxies or the client. For each segment, the data server scans the P-proxies according to the content service path; if a P-proxy needs to read this segment, then the data server adds this P-proxy to its succ, and this P-proxy will receive this segment from the data server. The data server repeats this activity until the first P-proxy that needs to update this segment. After the P-proxy updates the segment, it sends out the segment to the rest of the P-proxies or the client if they need to access it.



**Fig.3.5. Data Server**

# TECHNOLOGY INFRASTRUCTURE

## 4.1 CORE JAVA

Java can be used to create two types of programs: application and applet. An application is a program that runs on your computer, under the operating system of that computer. That is, an application created by java is more or less like one created using C or C++. When used to create application, java is not much different from any other computer language. Rather, it is java's ability to create applets that makes it important. An applet is an application designed to be transmitted over the internet and executed by a java-compatible Web Browser. An applet is actually a tiny java program, dynamically downloaded across the network, just like an image, sound file, or video clip. The important difference is that an applet is an intelligent program, not just an animation or media file. In other words, an applet is a program that can react to user input and dynamically change-not just run the same animation or sound over and over.

Java having a major role in internet and the intranet application. The reason for this is quite simple: Java expands the universe of objects that can move about freely in cyberspace. In a network, two very broad categories of objects are transmitted between the server and your personal computer: passive information and dynamic, active programs.

### **Security**

As you are likely aware, every time that you download a "normal" program, you are risking viral infection. Prior to java, most users did not download executable programs frequently, and those who did scan them for viruses prior to execution. Even so, most users still worried about the possibility of infecting their system with a virus. When you use a java-compatible web browser, you can safely download java applets without fear of viral infection or malicious intent.

Java achieves this protection by confining a java program to the java execution environment and not allowing it access to other parts of computer.

many are connected to the internet. For program to be dynamically downloaded to all the various type of platforms connected to the Internet, some means of generating portable executable code is needed.

### **Bytecode**

The key that allows java to solve both the security and the portability problems just described is that output of a java compiler is not executable code. Rather, it is BYTECODE. Byte code is a highly optimized set of instruction designed to be executed by the java runtime system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreted code.

### **Simple**

Java was designed to be easy for the professional programmer to learn and use effectively. Assuming that you have some programming experience, you will not find java hard to master. If we know the basic concept of object-oriented programming, learning java will be even easier.

### **Object-Oriented**

Object-Oriented programming is the core of java. In fact, all java programs are object-oriented-this isn't an option the way that it is in C++, for example. OOP is so integral to java that you must understand its basic principles before you can write even simple java programs.

### **Abstraction**

The essential element of object-oriented programming is abstraction. Humans manage complexity through abstraction. For example, people do not think of a car as a set of ten of individual parts. They think of it as a well-defined object with its own unique behavior. So this ignore the details of how the engine, transmission, and braking systems work.

## **4.2 SWING**

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit. Swing provides a native look and feel that emulates the

Swing introduced a mechanism that allowed the look and feel of every component in an application to be altered without making substantial changes to the application code. The introduction of support for a pluggable look and feel allows Swing components to emulate the appearance of native components while still retaining the benefits of platform independence. This feature also makes it easy to make an application written in Swing look very different from native programs if desired. Originally distributed as a separately downloadable library, Swing has been included as part of the Java Standard Edition since release 1.2. The Swing classes and components are contained in the `javax.swing` package hierarchy.

## **Architecture**

Swing is a platform-independent, Model-View-Controller GUI framework for Java. It follows a single-threaded programming model, and possesses the following traits:

### **Platform independence**

Swing is platform independent both in terms of its expression (Java) and its implementation (non-native universal rendering of widgets).

### **Extensibility**

Swing is a highly partitioned architecture, which allows for the "plugging" of various custom implementations of specified framework interfaces: Users can provide their own custom implementation(s) of these components to override the default implementations. In general, Swing users can extend the framework by extending existing (framework) classes and/or providing alternative implementations of core components.

### **Component-oriented**

Swing is a component-based framework. The distinction between objects and components is a fairly subtle point: concisely, a component is a well-behaved object with a known/specified characteristic pattern of behaviour. Swing objects asynchronously fire events, have "bound" properties, and respond to a well-known set of commands (specific to

## Customizable

Given the programmatic rendering model of the Swing framework, fine control over the details of rendering of a component is possible in Swing. As a general pattern, the visual representation of a Swing component is a composition of a standard set of elements, such as a "border", "inset", decorations, etc. Typically, users will programmatically customize a standard Swing component (such as a JTable) by assigning specific Borders, Colors, Backgrounds, opacities, etc., as the properties of that component. The core component will then use these property (settings) to determine the appropriate renderers to use in painting its various aspects. However, it is also completely possible to create unique GUI controls with highly customized visual representation.

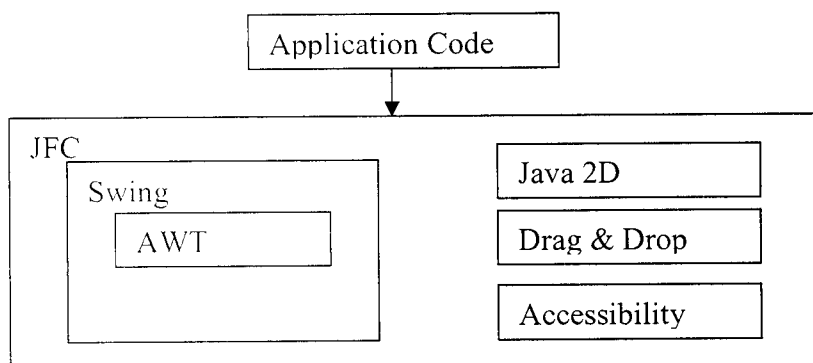


Figure 4.1 Swing class

## 4.3 ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the



Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program, and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the

## 4.4. JDBC

In an effort to set an independent database standard API for Java, Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

### JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The eight design goals for JDBC are as follows:

1. ***SQL Level API***

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

2. ***SQL Conformance***

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through

3. ***JDBC must be implemental on top of common database interfaces***

The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

4. ***Provide a Java interface that is consistent with the rest of the Java system***

Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

5. ***Keep it simple***

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

6. ***Use strong, static typing wherever possible***

Strong typing allows for more error checking to be done at compile time; also, less errors appear at runtime.

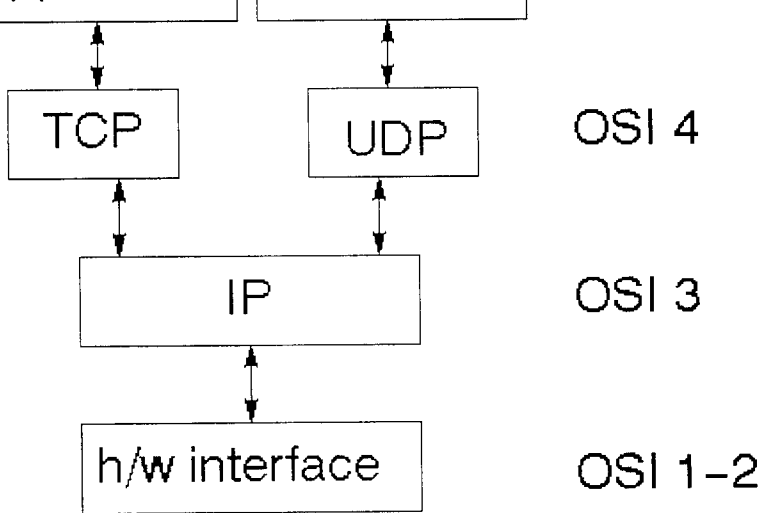
7. ***Keep the common cases simple***

Because more often than not, the usual SQL calls used by the programmer are simple SELECT’s, INSERT’s, DELETE’s and UPDATE’s, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

## **4.5. Networking**

### **TCP/IP stack**

The TCP/IP stack is shorter than the OSI one:



**Fig.4.2. TCP/IP Stack**

TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

### **IP datagram's**

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

### **TCP**

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

### **Internet addresses**

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives

## Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

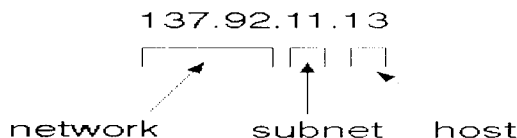
## Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

## Host address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

## Total address



**Fig.4.3. IP Addressing**

The 32 bit address is usually written as 4 integers separated by dots.

## Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

A socket is a data structure maintained by the system to handle network connections.

A socket is created using the call `socket`. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with `Read File` and `Write File` functions.

```
#include <sys/types.h>
```

```
#include <sys/socket.h>
```

```
int socket(int family, int type, int protocol);
```

Here "family" will be `AF_INET` for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

## EXPERIMENTAL RESULTS

## 5.1. CLIENT AUTHENTICATION



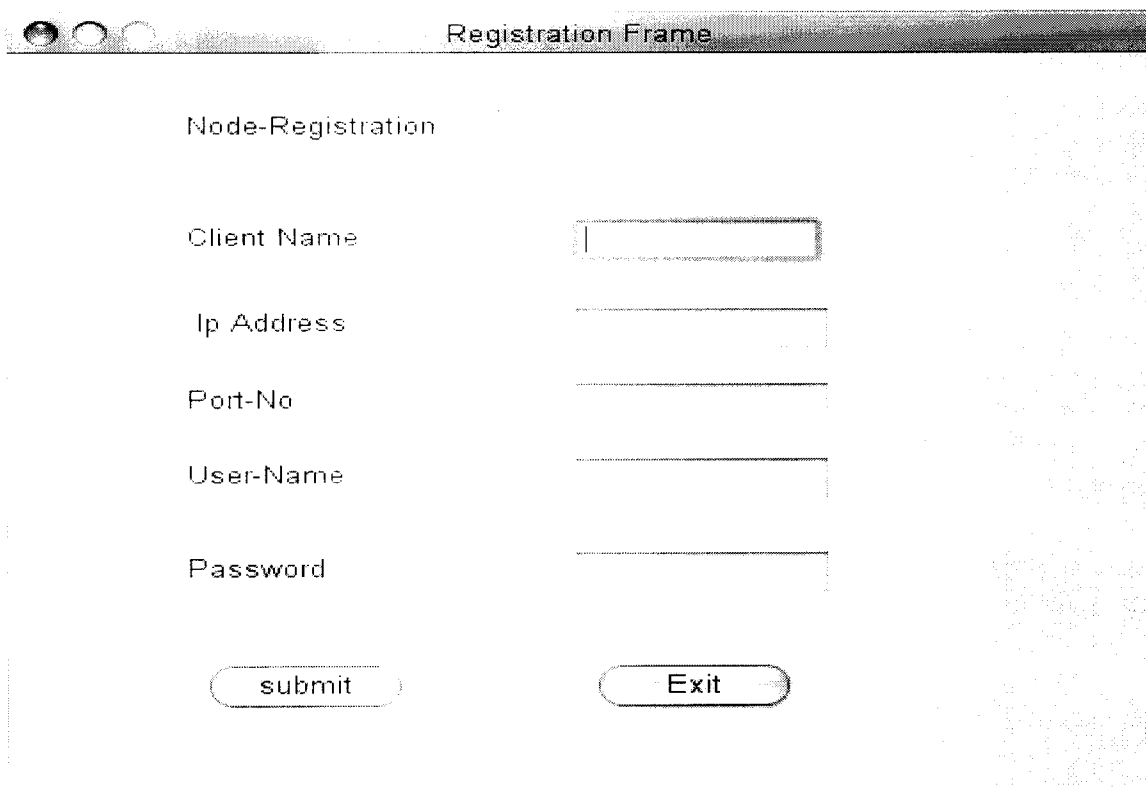
The image shows a screenshot of a web browser window titled "LoginPage Frame". The window contains a form titled "Authentication". The form has two input fields: "User Name" and "Password". Below the "User Name" field is a "Sign-Up" button, and below the "Password" field is a "Sign-In" button. The buttons are rounded rectangles with a slight shadow effect.

**Authentication**

User Name

Password

## 5.2. CLIENT REGISTRATION



The image shows a graphical user interface window titled "Registration Frame". The window contains a form for client registration. The form has five input fields, each with a label to its left: "Client Name", "Ip Address", "Port-No", "User-Name", and "Password". Below the input fields are two buttons: "submit" and "Exit".

Node-Registration

Client Name

Ip Address

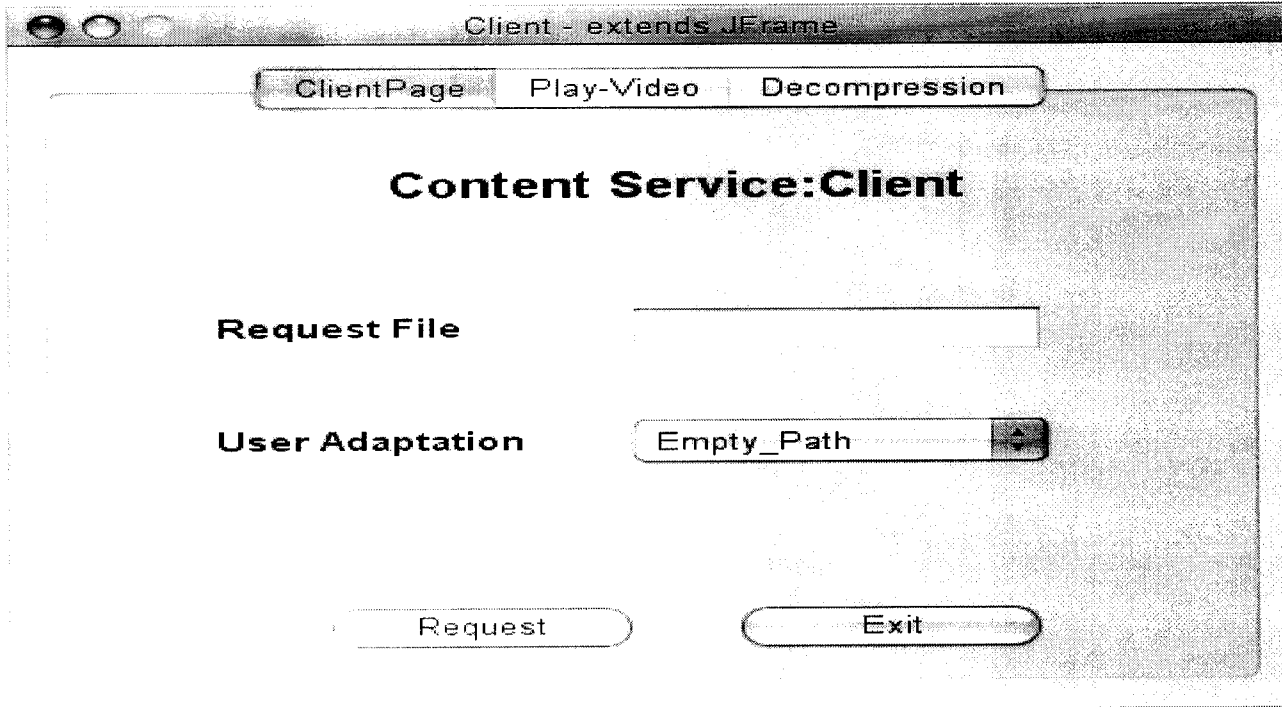
Port-No

User-Name

Password



## 5.3. CLIENT VIEW



## Content Service:Client

**Request File**

hi.txt

**User Adaptation**

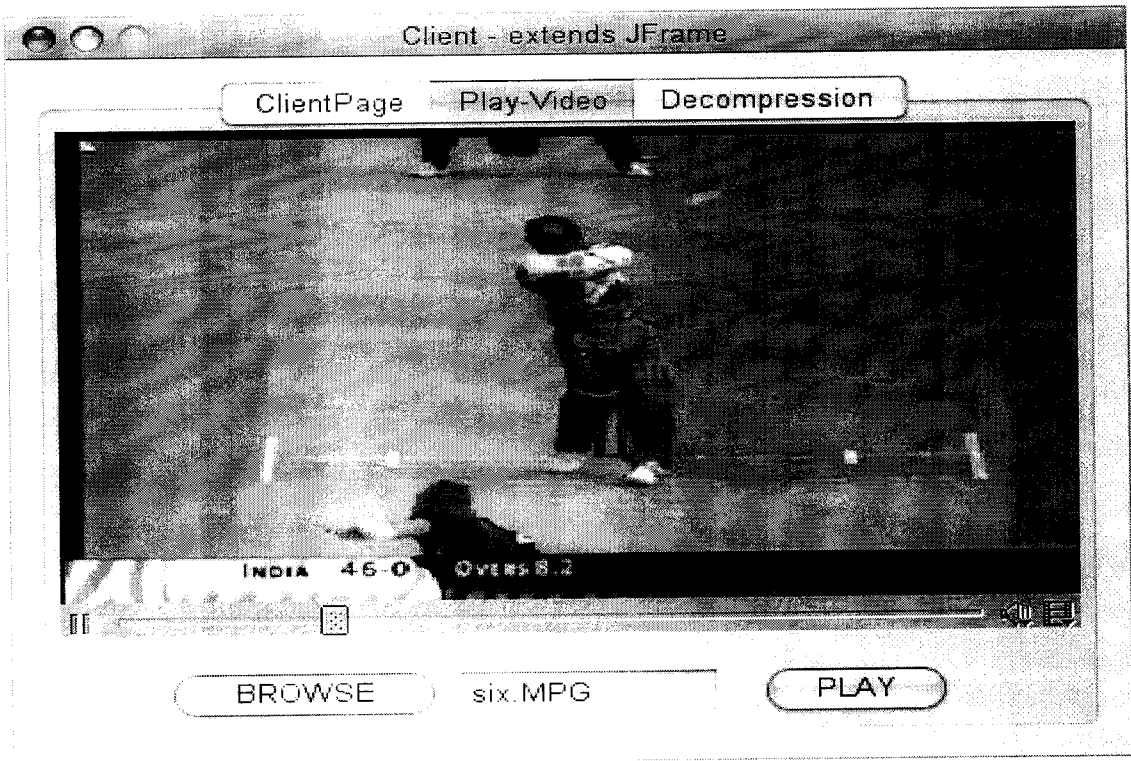
Empty\_Path

Downloading 60%

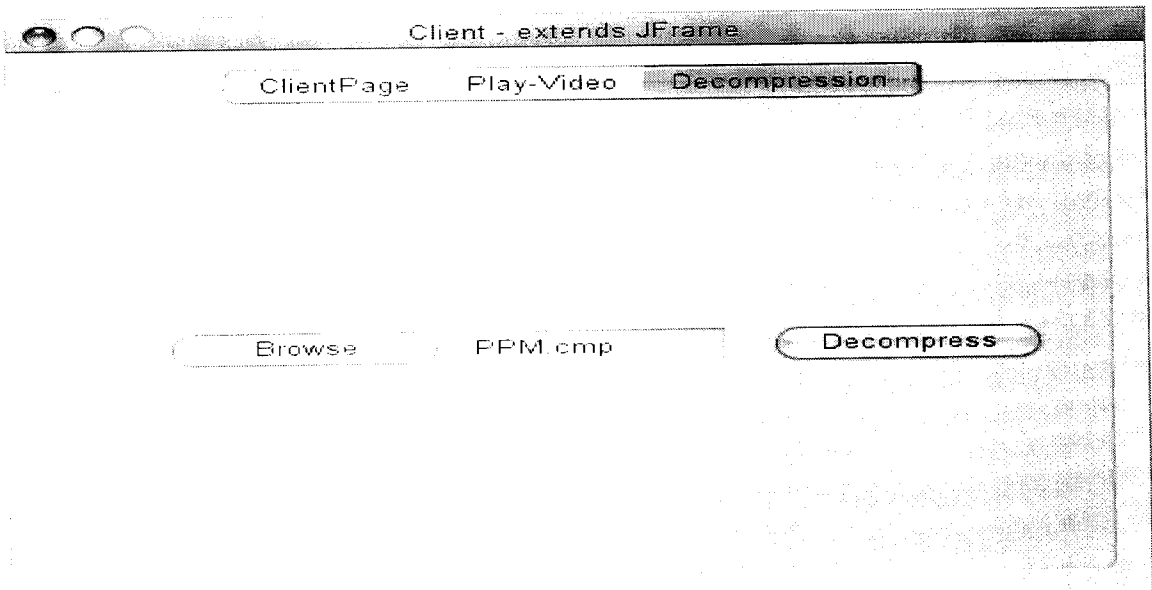


Request

Exit



## 5.6. DECOMPRESSION



Data Server

File List

6.jpg  
ajith1.jpg  
ajith2.jpg  
ajith3.jpg  
hrithik1.jpg  
hrithik2.jpg  
suri.jpg  
Thumbs.db  
PPM.jpg

File Types

- Image\_Files
- Pdf\_Files
- Text\_Files
- Videos

Request Details

UserName:rafik	IPAddress:bct-155/192.168.1.18	File Name:six.MPG
UserName:rafik	IPAddress:bct-155/192.168.1.18	File Name:Shahid A
UserName:rafik	IPAddress:bct-155/192.168.1.18	File Name:Shahid A
UserName:rafik	IPAddress:bct-155/192.168.1.18	File Name:Shahid A

SQL Query Analyzer

File Edit Query Tools Window Help

contentservice

Query - BCT-155.contentservice.sa - Untitled1\*

```
select * from userinformation
```

	clientname	ipaddress	portno	username	pwd
1	arun	localhost	4044	arun	Juju
2	nid	localhost	4444	arun	kumar
3	sen	localhost	4000	sen	sen
4	sendhil	localhost	4004	dhil	dhil
5	nid	localhost	4444	nid	mar
6	yy	localhost	9999	yy	yy
7	zz	localhost	5478	zz	zz
8	rr	localhost	5555	rr	rr

Grids Messages

Query batch completed. BCT-155 (8.0) sa (52) contentservice 0:00:00 23 rows Ln 1, Col 30

Connections: 1 NUM

# CONCLUSION AND FUTURE OUTLOOK

## 6.1 CONCLUSION

This approach allows a client to verify that the received data is authentic and transformations on the data are properly authorized. It also assures data confidentiality during transmission. It also highlights load distribution through proxies to improve system performance and supports parallel content services. Because no modification is required to current content distribution systems in order to adopt this approach, also it is easy to deploy for many applications. In addition, this approach is extensible; if a new type of content service is required, this architecture can be easily adapted to the new requirement.

## 6.2 FUTURE OUTLOOK

No modification is required to current content distribution systems in order to adopt this approach, this work is easy to deploy for many applications. In addition, this approach is extensible; if a new type of content service is required, this architecture can be easily adapted to the new requirement.

## DATA SERVER

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
import java.rmi.NotBoundException;
import java.io.PrintWriter;
import java.io.IOException;
import java.io.BufferedReader;
import java.io.FileReader;

public class DataServer_Client
{
    private PrintWriter pw;

    private String systemName;

    private RemoteFile objRemoteFile;

    public DataServer_Client()
    {
        try
        {
            BufferedReader br = new BufferedReader( new
FileReader("dataserver.txt") );

            systemName = br.readLine().trim();

            br.close();

            pw = new PrintWriter( "fileinfo/filenames.txt" );
        }
    }
}
```

```

        ioe.printStackTrace();
    }
}

public String getFileNames()
{
    String strFileNames = "";

    try {

        objRemoteFile = ( RemoteFile )
            Naming.lookup(
                "rmi://" + systemName + "/FileService");

        System.out.println( "hello" );

        strFileNames = objRemoteFile.getFileNames();

        System.out.println( "FileNames:" + strFileNames );

    }
    catch ( MalformedURLException murle )
    {
        System.out.println(
            "MalformedURLException");
        System.out.println(murle);
    }
    catch ( RemoteException re )
    {
        System.out.println();
        System.out.println(

```



```

        }
        catch (NotBoundException nbe)
        {
            System.out.println();
            System.out.println(
"NotBlankException");
            System.out.println(nbe);
        }

        return strFileNames;
    }

    public void fileNames( String strFileNames )
    {
        String strArray[ ] = strFileNames.split( "&" );

        for( int i = 0; i < strArray.length; i++ )
        {
            pw.println( strArray[i] );
        }

        pw.flush( );
    }

    public byte[] getFile( String fileName )
    {
        byte byteArray[] = null;

        try
        {

```

```

        System.out.println( "byte" + byteArray );

        System.out.println( "byte array length" + byteArray.length );
    }
    catch ( RemoteException re )
    {
        re.printStackTrace( );
    }

    return byteArray;
}

// public static void main( String... a )
// {
//     DataServer_Client objDataServer = new DataServer_Client();
//
//     new BeanDataServerClient( ).setDataServer_Client( objDataServer );
//
//     objDataServer.fileNames( objDataServer.getFileNames( ) );
// }
}

```

## **INTERMEDIARY1**

```

package proxy2;

import java.util.zip.*;

import java.io.*;

```

```

public class ZIP_FILE
{
    private Proxy_Checking objProxy_Checking;

    public ZIP_FILE( )
    {
        objProxy_Checking = new Proxy_Checking();
    }

    public byte[] toZIP( String fileName, byte[] byteFileArray )
    {
        // Create a buffer for reading the files
        byte[] buf = new byte[1024];

        String[] strSplit = fileName.split( "\\." );

        objProxy_Checking.clearMemory( "proxy2\\cache" );

        try
        {
            File file = new File( fileName );

            FileOutputStream fos = new FileOutputStream( file );

            fos.write( byteFileArray, 5, byteFileArray.length-5 );

            fos.close( );

            ZipOutputStream out = new ZipOutputStream( new
FileOutputStream( "proxy2\\cache\\"+strSplit[0]+". "+"zip" ) );

```

```

        // Add ZIP entry to output stream.
        out.putNextEntry( new ZipEntry( fileName) );

        // Transfer bytes from the file to the ZIP file
        int len;

        while ( ( len = in.read( buf ) ) > 0)
        {
            out.write( buf, 0, len );
        }

        // Complete the entry
        out.closeEntry();

        in.close();

        // Complete the ZIP file
        out.close();

        file.delete( );

    }
    catch( FileNotFoundException fnf )
    {
        fnf.printStackTrace( );
    }
    catch ( IOException ioe )
    {
        ioe.printStackTrace( );
    }
    return getByteArray( strSplit[0] );
}

```

```

    {
        byte[] byteArray = null;

        try
        {
            FileInputStream fis = new FileInputStream(
"proxy2\\cache\\"+fileName+". "+ "zip" );

            byteArray = new byte[ fis.available() + 5 ];

            System.arraycopy( "Rfile".getBytes( ), 0, byteArray, 0, 4 );

            fis.read( byteArray, 5, byteArray.length-5 );

            fis.close( );

        }
        catch( FileNotFoundException fnf )
        {
            fnf.printStackTrace( );
        }
        catch ( IOException ioe )
        {
            ioe.printStackTrace( );
        }

        return byteArray;
    }

public static void main( String... args )
{
}

```



1. C. Aggarwal, J.L. Wolf, and P.S. Yu, "Caching on the World Wide Web," IEEE Trans. Knowledge and Data Eng., vol. 11, no. 1, pp. 94-107, Jan. 1999.
2. G.Berhe, L.Brunie, and J.M. Pierson, "Modeling Service-Based Multimedia Content Adaptation in Pervasive Computing," Proc. First Conf. Computing Frontiers, Apr. 2004.
3. L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," Proc. IEEE INFOCOM '99, Mar. 1999.
4. S. Buchholz and A. Schill, "Adaptation-Aware Web Caching: Caching in the Future Pervasive Web," Proc. 13th GI/ITG Conf. Kommunikation in Verteilten Systemen (KiVS), 2003.
5. V. Cardellini, P.S. Yu, and Y.W. Huang, "Collaborative Proxy System for Distributed Web Content Transcoding," Proc. Ninth ACM Int'l Conf. Information and Knowledge Management (CIKM '00), Nov. 2000.
6. S.Chandra and C.S. Ellis, "JPEG Compression Metric as a Quality- Aware Image Transcoding," Proc. Second Usenix Symp. Internet Technology and Systems (USITS '99), Oct. 1999.
7. C.H.Chi, Y.Lin, J.Deng, X.Li, and T.Chua,"Automatic Proxy- Based Watermarking for WWW," Computer Comm., vol. 24, no. 2,pp. 144-154, Feb. 2001.
8. C.H.Chi and Y.Wu, "An XML-Based Data Integrity Service Model for Web Intermediaries," Proc. Seventh Int'l Workshop Web Content Caching and Distribution (WCW '02), Aug. 2002.
9. Extensible Markup Language (XML), <http://www.w3.org/XML/,007>.

Comm., Aug. 1998.

11. M.J. Freedman, E.Freudenthal, and D.Mazie`res, "Democratizing Content Publication with Coral," Proc. Usenix/ACM Symp. Networked Systems Design and Implementation (NSDI '04), Mar.2004.