# The New Convergence for Similitude Examination of Data

Bhanu Rekha Pydi[1] D.T.V. Dharmajee Rao[2]
[1](Final Year M.Tech Student, Dept. of CSE, Aditya Institute of Technology and Management (AITAM), Tekkali, Srikakulam, Andhra Pradesh, bhanu6996@gmail.com)
[2](Professor, Dept. of CSE, Aditya Institute of Technology and Management(AITAM), Tekkali, Srikakulam, Andhra Pradesh,dharmajee@hotmail.com)

## *Abstract:*

To achieve similitude examination of data, first accumulate the data in cloud. Due to the appealing features of cloud computing, accumulation of more data is possible. But the security of data is big concern. So the encrypted and compressed accumulation protects the data from illegal access. To do this, use BASE-64 and Zip compression algorithms. After accumulation simply cleans the data and categorizes the data. When the user examines the data most related information to be retrieved. To do these use the data sensitivity technique by using DSI (Data Sensitivity and Insensitivity) algorithm. Also examination of data can be done in simple manner using LDAP (Light Weight Directory Access protocol). It is used to control and organizing the data in cloud. Finally, when the user sends the query the most related data can be retrieved effectively.

## *Index terms:*

Data cleaning, data categorization, data accumulation, data sensitivity, LDAP (Light Weight Directory Access Protocol), data insensitivity, encrypted and decrypted data, compression and decompression.

## *Introduction:*

The cloud accumulates more data and Moving data into the cloud offers great convenience to users since they don't have to care about the complexities of direct hardware management. But the security and privacy is the main problem in cloud.

The accumulation of data in un-trusted cloud gives the security problem. So to overcome the security problem accumulate the encrypted and compressed data in cloud .it prevents the unauthorized access and reduces the size of the data.

Once the client accumulates the data in cloud then users can access the required data by sending a query to server (cloud). When we categorize the data the searching process can be done quickly.

In the cloud we are maintaining two devices. One device is used to accumulate the data and another device is used to control the accumulated data. The accumulated device maintains the sensitive and in-sensitive data. We consider a threshold value. If any document exceeds the threshold value then the data is considered as sensitive data and less than the threshold value considered as in-sensitive data. The control device is used to control the accumulated data. The control device consists of LDAP protocol. When the user sends a query to server the LDAP protocol take's the responsibility to send the required related data to user. The control device consists of folders. Each folder contains related files. So based on the threshold value the LDAP protocol examine the data. So, LDAP will play vital role at the service side. This LDAP will maintain all the information of the accumulated data. This information

helps to save the time for search criteria. So when user sends a request the most related data to be retrieved.

## *Client-Server Architecture:*

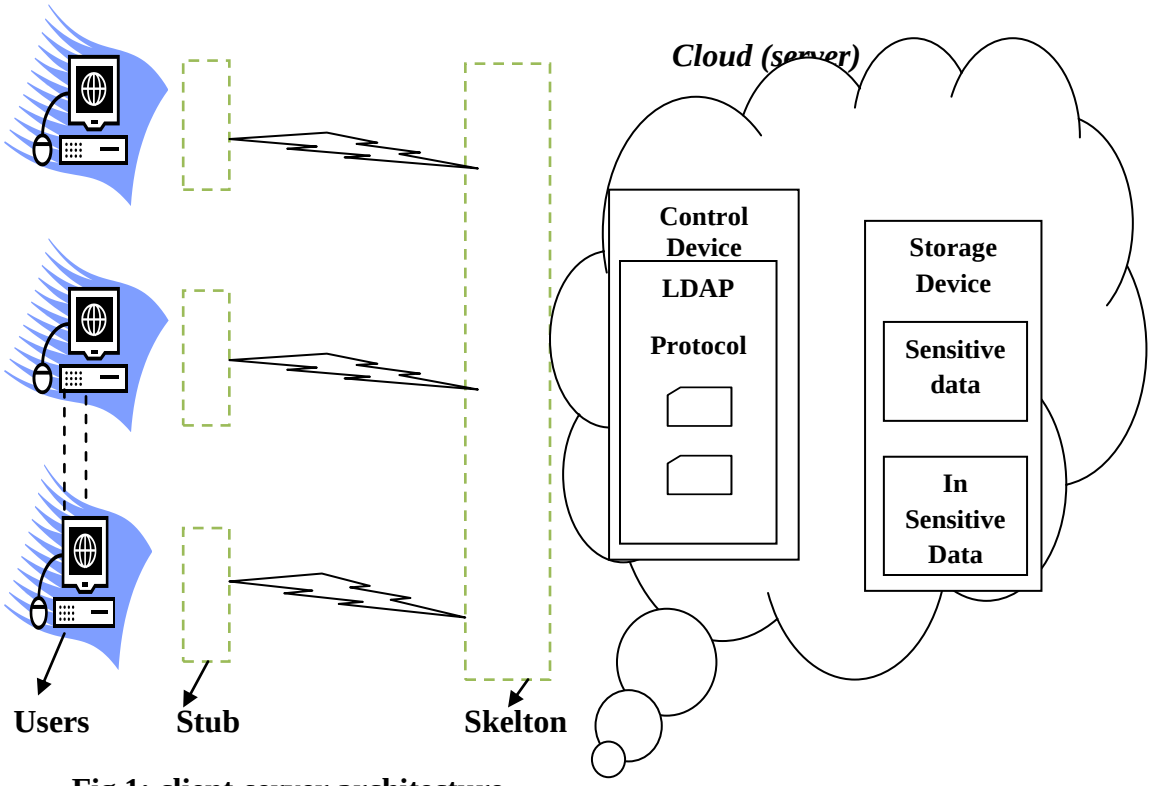


**Fig 1: client-server architecture**

When the client accumulates the encrypted and compressed data in cloud then the users can access the required data from cloud. Here cloud is treated as server. The users send a query to server to get the required information. The server consists of a control device. The control device maintains LDAP (Light Weight Directory Access Protocol). When the server receives the request from users it passes to LDAP protocol. The LDAP protocol examines the data from storage device. First it checks the data based on threshold value. After examination simply it collects the related data and forwards to server. The server transmits the data to user. In summary, there are several notable contributions of this paper. They are,

***Data Cleaning:*** It is a technique to remove the unnecessary data. The data consists of UTF characters. If the data consists of unnecessary symbols or data then the examination of data is difficult. So first remove the unnecessary data from cloud.

***Data categorization:*** categorizing the data is nothing but place the related data in one folder. In the same way we separate all data Put it in related folders. The data categorization simplifies the examination of data.

***Data Sensitivity and In-Sensitivity:*** After performing data categorization, we examine the sensitivity of data. Consider a threshold value. If any document exceeds the

threshold value then the data is considered as sensitive data and the remaining data is considered as in-sensitive data. Here data sensitivity means that information is mostly related to user's request.

***Data Encryption:*** The data encryption is applied on the categorized data. The encrypted accumulation protects the data from un-authorized access. We are using base-64 algorithm to encrypt the data.

***Data compression:*** To perform the data compression technique on the encrypted data use ZIP compression. It reduces the size of the data in cloud.

***Data decryption and decompression:***
When the user sends a query to server before the searching process the data is decrypted and decompressed and then the required data is sends to the user.

***Examine the data based on user's request:***
When the client accumulates the data in cloud then the data is shared to permitted data users. So if any user wants to use the data then the user sends a query to server. The query consists of several words and each word is considered as an object. The query in each word is compared with the data accumulated in cloud. If any data is suitable to the particular requested query then that data is retrieved based on the threshold value and it is considered as sensitive data. That is the data is mostly related to users request and the data is send to the user.

***LDAP:*** LDAP stands for Lightweight Directory Access Protocol. It is a protocol for accessing directories originally designed to act as a gateway to other directories. It is an application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network. It placed in server side to manage the accumulated data.

***Algorithm1:*** Data categorization

In this algorithm first clean the data and categorize the data. Based on the categorized data perform data sensitivity calculation.

***Steps:*** Initially the cloud consists of number of non cleaned documents (Nc).

Consider each document to clean the data.

1. It removes the UTF characters from each document one by one. So examination of data can be done easily.
2. After cleaning categorize the data. Here each document is checked with related category like c, .net, java etc.
3. If any document is related to one category then the document not compared with remaining categories.
   EX: if the second document is matched with java category then the document not checked with remaining categories.
4. Finally we get the categorized data. Next we have to perform data sensitivity calculation.

***Terminology:***

$N_C$: Number of non cleaned documents,

$\sum_0^{n-1}$ C: Data categorization,

$\sum_0^{n-1}$ V: UTF characterized vector,

$\sum_0^{n-1} N_S$: Non sensitive vector,

$\sum_0^{n-1}$ $S_V$: Sensitive vector,

$\sum D_C$: Data vector(s) to mine,

$T_C$: Temporary document vector,

$CT_C$: Temporary document category vector,

$\sum$ P: Temporary document vector

***Data categorization:***

Input → $N_C$

Output → P

Count=0

For each d (document) in $N_C$

$T_C \in$ UTFR (d)

Count++

End for

For each category c in C

For each d (document) in $T_C$

If d ∈ c

$CT_C$ ← d

P ← $CT_C$

$CT_C$ – d

End if

End for

***Algorithm2:*** DSI (Data Sensitivity and Insensitivity Calculation)

***Steps:***

1. We give categorized data as input to perform data sensitivity calculation.
2. Initially we take threshold value. For each categorized document we assign fixed threshold value.
3. If any categorized document exceeds the threshold value then the document can be considered as sensitivity data. Otherwise the document can be considered as insensitivity data.
4. The data sensitivity calculation is performed to retrieve most related data based on user's request.

***Data Sensitivity and Insensitivity Calculation:***

Input → P

Output → $D_C$

Initialization $\theta$=15

Loop statement

For all c in C

For each p in P

If $0 \leq P_{count}$ (p) ≤15

$D_C \in$ Mark (p)

Else

$D_C$ ← p

End if

End for

End for

End loop

## *Algorithm3:* Base64

***Encryption:*** Base64 is a group of similar binary-to-text encoding schemes that represent binary data in an ASCII string format by translating it into a radix-64 representation.

Base64 encoding schemes are commonly used when there is a need to encode binary data that needs to be stored and transferred over media that are designed to deal with textual data. This is to ensure that the data remain intact without modification during transport. Base64 is commonly used in a Base64 encoding takes the original binary data and operates on it by dividing it into tokens of three bytes. A byte consists of eight bits, so Base64 takes 24bits in total. These 3 bytes are then converted into four printable characters from the ASCII standard.

The algorithm's name Base64 comes from the use of these 64 ASCII characters. The ASCII characters used for Base64 are the numbers 0-9, the alphabets 26 lowercase and 26 uppercase characters plus two extra characters '+' and '/'.

The first step is to take the three bytes (24bit) of binary data and split it into four numbers of six bits. Because the ASCII standard defines the use of seven bits, Base64 only uses 6 bits (corresponding to 2^6 = 64 characters) to ensure the encoded data is printable and none of the special characters available in ASCII are used.

The ASCII conversion of 3-byte, 24-bit groups is repeated until the whole sequence of original data bytes is encoded. To ensure the encoded data can be properly printed and does not exceed the limit.

When the number of bytes to encode is not divisible by 3 (that is, if there are only one or two bytes of input for the last 24-bit block), then the following action is performed: Add extra bytes with value zero so there are three bytes, and perform the conversion to base64. If there was only one significant input byte, only the first two base64 digits are picked (12 bits), and if there were two significant input bytes, the first three base64 digits are picked (18 bits). '=' characters might be added to make the last block contain four base64 characters.

***Example:***

| ***Text content*** | ***M*** | ***a*** | ***n*** |
|---|---|---|---|
| ***ASCII*** | ***77*** | ***97*** | ***110*** |
| ***Bit pattern*** | ***0 1 0 0 1 1 0 1 0 1 1 0 0 0 0 1 0 1 1 0 1 1 1 0*** | | |
| ***Index*** | ***19*** | ***22*** | ***5*** | ***46*** |
| ***Base64-encoded*** | ***T*** | ***W*** | ***F*** | ***u*** |

***Padding:***

The '==' sequence indicates that the last group contained only 1 byte, and '=' indicates that it contained 2 bytes.

***Exapmple1:***

Input:
*any carnal plea**sure.***
Output:
YW55IGNhcm5hbCBwbGVhc3VyZS4=

***Exapmple2:***

Input:
*any carnal plea**sure***

Output:
YW55IGNhcm5hbCBwbGVh**c3VyZQ==**

***Base64 (encryption) algorithm:***

Input → raw string

Output → base64 encoded format

***Step1:*** **initialization**

$\int$ALPHABET = $\sum_0^{25}$ CAPS$(65-90)$ + $\sum_0^{25}$ SMALL$(97-122)$ + $\sum_0^{9}$ NUMBERS$(48-57)$ + $\sum_0^{2}$ SPC$(43,67)$

// all "ALPHABET" CONTAINS ASCII values for capital letters, small letters, single digit numbers, '+' and '/' characters.

***Step2:***

***Functionality:***

To convert alphabets to ASCII codes

Input ← all available characters

Output ← all equivalent ASCII values

n ← 0

B0 ← 0

B1 ← 0

B2 ← 0

$\sum_0^n$ buff[] ← 0

$\sum_0^n$ ar ← 0

**i** ← 0

Iteration ← 0

Loop statement:

Count=0

For each C in ALPHABET

toInt (count) = TOINT (ALPHABET (i))

Count++

End loop

Size=SIZE (buff)

Iteration ← (((size+2)/3)*4)

Do while n in iteration

B0 ← buff

B1 ← (i <size)? buff++: 0

B2 ← (i < size)? buff++: 0

***Masking***

Mask=0X3F

ar = ALPHABET [(b0>>2) & mask]

ar= ALPHABET [(b0<<4) | ((b1&0XFF)>>4)) & mask]

ar= ALPHABET [(b1<<2) | ((b2&0XFF)>>6)) & mask]

ar = ALPHABET [b2 & mask]

End while

**Padding:**

If (size % 3=1)

ar ← "="

ar ← "="

Else if (size % 3=2)

ar ← "="

else

size %3=0

End if

***Decryption:***

When decoding Base64 text, 4 characters are typically converted back to 3 bytes. The only exceptions are when padding characters exist. A single '=' indicates that the 4 characters will decode to only 2 bytes, while 2 '='s indicates that the 4 characters will decode to only a single byte.

***Example:***

Input:
YW55IGNhcm5hbCBwbGVh**cw==**
Block with 2 '='s decodes to 1 character:
Output:
*any carnal plea**s***

***Base64 (decoding) algorithm:***

Input ← string

Output ← $\sum_{0}^{n}$ buff(bytes)

***Initialization:***

Buff ← 0

S ← string decode

N ← length of string

Mask ← 0XFF

If s [0] = '=='

Delta =2

Else if s [0] = '='

Delta =1

Else

Delta=0

End if

Loop

For I ← 0 step by 4 of in n

C0=Convert to Int [CharAt (i) in S]

C1=Convert to Int [CharAt (i+1) in S]

$buffer_i$ ← (c0<<2) | (c1>>4) & mask

C2 ← Convert to Int [CharAt (i+2) in S]

$buffer_{i+1}$ ← (c1<<4)|(c2>>2)&mask

C3=convert toInt [i+3]

$buffer_{i++}$ = (c2<<6)! c3&mask

End loop

***Algorithm4:*** Compression and Decompression.

The compression technique is performed on encrypted data. We are using compression technique to reduce the size of the data. So we can store more data on cloud. We are using ZIP compression technique. A zip file is a single file that contains one or more compressed files. Zip files usually have the ".ZIP" filename extension, which helps to clearly identify them.

In cloud each file in a zip file consists of compressed data or as raw uncompressed data, and can be encrypted.

The zip file format is designed to support more than one type of compression method, so that new and improved compression methods can be used. This ZIP algorithm applicable to maximum size 1GB of data. It compress the more than 80% of data. When decompression can be done it gives the original data without loss.

***Compression Algorithm:***

Input: file

Output: compressed file (zip file)

Step 1:

Take a file from directory (along with path)

From ← file (input)

Specify directory to save zip file (path)

To ← file (output)

Step 2:

Initialize buffer with specified size

Buffer ← byte [4096]

Step 3:

Initialize bytes read ← buffer

Buffer1 ← read (from)

Bytes read ← read (buffer)

Loop starts

While bytes read! = -1

Buffer ← write

Step 4:

Create zip directory (dir, zip file)

If dir! =available directory then

Directory not available

Step 5:

Buffer for copying

Buffer ← byte [4096]

Entries ← data in buffer

Step 6:

Loop starts for entries in buffer

For all entries in buffer

I ∈ entries.length

File ← add (entries [I])

To ← zip (file)

d ← mk.dir (to)

Save (d)

***Decompression:***

Step 1:- input: Zip file.

Consider the zip file path with removing the (.Zip 4 characters) Extension.

Temp ← path

Mk.dir (temp)

Step 2 :- Read content

Content ← zip file. Entries

Loop start // entries in file.

File.entries← iterate each entry from zip File,

Dest File ← File (path, file.entries)

Directory ← Mk.dir (Dest File)

Step3:-

If available directory then

Save File into directory

Else

Buffer ← bytes [1024]

Buffer ← [Read the current directory from zip file and extract]

Loop starts //

While buffer! = -1

Write (buffer)

(Write Extracted File.)

## *Experimental results:*

The experimental results shows that when user sends a request based on threshold value the related data was retrieved. If client fix large number of threshold value then less number of data but most related to requested data was retrieved. If the threshold value is less then more related information related to request was retrieved. So based on threshold value the retrieved data was varied.

***Conclusion:*** In this paper, we proposed new convergence for similitude examined data. The similitude examined data was retrieved when the user requested the server. Based on threshold value sensitivity of data was calculated on categorized data. So the result shows that fast and effective similitude examined data. So when the user sends a query the most related data can be retrieved based on threshold value.

## *References:*

[1] Mehmet Kuzu, Mohammad Saiful Islam, Murat Kantarcioglu,"Efficient Similarity Search over Encrypted Data".

[2] J. Li, Q. Wang, C. Wang, N. Cao, K.Ren, and W. Lou, "Enabling efficient fuzzy keyword search over encrypted data in cloud computing," in *Cryptology ePrint Archive, Report 2009/593*, 2009.

[3] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probelsh: Efficient indexing for high-dimensional similarity search," in *Proc.of VLDB'07*, 2007, pp. 253–262.

[4] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. of VLDB'99*, 1999, pp. 518–529.

[5] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. http://infolab.stanford.edu/ullman/mmds/book.pdf, 2010.

[6] C. Faloutsos and K. Lin, "Fastmap: a fast algorithm for indexing, data mining and visualization," in *Proc. of the SIGMOD'95*, 1995, pp. 163–174.

[7] Federal Cloud Computing Strategy, http://www.cio.gov/documents/Federal-Cloud-Computing-Strategy.pdf

[8] Chief Information Officers Council, "Privacy Recommendations for Cloud Computing", http://www.cio.gov/Documents/Privacy-

Recommendations-Cloud-Computing-8-19-2010.docx

[9] NIST SP 800-144, "Guidelines on Security and Privacy Issues in Public Cloud Computing", http://csrc.nist.gov/publications/drafts/800-144/Draft-SP-800-144_cloud-computing.pdf

[10] IETF internet-draft, "Cloud Reference Framework", http://tools.ietf.org/html/draft-khasnabish-cloud-reference-framework-00