A Secured removal code based cloud storage system with safe records a head

[1]Polaki Tarakeswara Rao M.Tech, e-Mail: tarak.polaki@gmail.com
Guide Name: Sri. M.V.B Chandra Sekhar, Assoc. Prof.
AITAM, Tekkali

Abstract: Basically services will be having the storage areas where the clients can throw and preserve the data in bulk storage. To encrypt the data we propose the CCFE algorithm coz some other algorithms will also encrypt the data but not up to the level of 100% encryption coz of some strange formats of the data. Though the encryption is happening clients can directly interact with services in a distributed way. So this is very sensitive to services to restrict the confidentiality. So root services should be having the authority hierarchy to decrypt according to the users/clients categories. So we propose CCFE and without any further queries like percentage of compression, without any loss of data further in the distributed form. So the confidentiality architecture will be maintained throughout the hierarchy. The log will be maintained by the services, how many requests and types of requests coming from various clients. Once the client(s) started sending the request the monitor will monitor all the request cycle till it reach to service side. So all the request process will be monitored in the proper log file at stub side. The whole architecture is based on SOA (Service Oriented Architecture), where clients can have continuous services from service. By encrypting the data and the key will be sealed into the request data. After that the data will be encrypted. So the security will be maintained through clients' stub and service skeleton level (proxy structure). At service side the service will not have direct interaction, the reason is the security. Either stub or some intermediately program will accept the client's request and that will be redirected to service. So service will process the request and the request which is processed at the service in secured way.

(*Index terms: SOA, confidentiality, stub, proxy*)

Introduction: We are proposing a new SOA architecture in this paper. The regular way is client(s) can directly interact with service as this is very sensitive and the client can be intruder to hack/capture the sensitive information from service. And ancient cloud services will be having direct interaction though they have access to third party services. But in this architecture we propose an LDAP request dispatcher to route the requests to valid service indirectly. So the request will not go directly to service.

LDAP (Light weight directory access protocol is the protocol which allows in this architecture to route the calls to valid services(own services or third party services). In this work LDAP

receives the request at skeleton side and does categorization to send to the particular service.

The asked for information will end up being encrypted/ the important thing will likely are invisible and also routed to help solutions facet. Below the important thing will likely be saved by the tracker to bring this response from assistance to normal form. And so all of us place magic formula information structure on customer facet to help keep this recommendation that happen to be random (unique) created and you will be in contrast to came back critical from assistance. In the event that that will matches the information will likely be presented to normal form to help customers.

The requested data also will be encrypted/ the key will be concealed and routed to services side. Here the key will be saved by the tracker to bring the response from service to normal form. So we put secret data structure at client side to maintain the keys which are random (unique) generated and will be compared with returned key from service. If that matches the data will be presented to normal form to clients.

Related work: Normally SOA(Service Oriented Architectures) are so strong in maintaining the concurrent client(s) as a services. Once the data is sent by client as request, the request is received by the LDAP and encrypts. This encrypted form is then passed to actual service it has to go. Then the service decrypts the encrypted form and sends it to the service.

Encoding is the process of putting a sequence of characters into a special format for transmission or storage purposes

Encryption is the process of translation of data into a secret code. Encryption is the most effective way to achieve data security. To read an encrypted file, you must have access to a secret key or password that enables you to decrypt it. Unencrypted data is called plain text ; encrypted data is referred to as cipher text

Encoding:

1. Purpose: The purpose of encoding is to transform data so that it can be properly (and safely) consumed by a different type of system.
2. Used for: For maintaining data usability i.e., to ensure that it is able to be properly consumed.
3. Data Retrieval Mechanism: No key and can be easily reversed provided we know what algorithm was used in encoding.
4. Algorithms Used: ASCII, Unicode, URL Encoding, Base64
5. Example: Binary data being sent over email, or viewing special characters on a web page.

Encryption:

1. Purpose: The purpose of encryption is to transform data in order to keep it secret from others.
2. Used for: For maintaining data confidentiality i.e., to ensure the data cannot be consumed by anyone other than the intended recipient(s).
3. Data Retrieval Mechanism: Original data can be obtained if we know the key and encryption algorithm used.
4. Algorithms Used: AES, Blowfish, RSA

Example: Sending someone a secret letter that only they should be able to read, or securely sending a password over the Internet.

LDAP: Light weight Directory Access Protocol will play key role in this work. Regular data will be stored in segregated directories whether they are in client side or service side. In this work the client will send a request for particular topic which has to be routed to valid directory where the data is saved and that data can local service or remote service. As in our work there is no direct interaction with the service LDAP will segregate that request and route to valid service to send back the response to client.

Terminology:

S_i ← Main Service

r_d ← Request Dispatcher

r_h ← request Handler

$\sum_0^n C$ ← Total No. of clients

t ← Tracker(Data Structure)

S_d ← Data source

K ← Key data structure

$\sum_0^m R$ ← Set of all requests to client

$\sum RS$ ← Set of all responses

k ← Random Key

Service creation: (Skelton)

Initialization

S_d ← 0

K_d ← 0 // where set of all keys will be observed

I ← index

For all services s in S

Loop

λ ← 50

S ←s

S_d (s) = Space (λ)
// space for each service in

i++

Data ←S ∑ Access(s,i)

R_d = data will be

To request description to send data re

S← data

End loop

Request Dispatches

Input: request/response

Output: reduction

Loop for each r in R || RS

Count = 0

If r € R

R=Modify (r)

Else if r € R_s

R_s = Modify (r, C_i)

End Loop

Client Callers (Stub)

Loop: for each c in C

c = { r1,r2,r3…….. r_n} → $\sum_0^n r$

SEND (r_d,r)

ENCR (r,k)

R ← r

K ← k

End loop

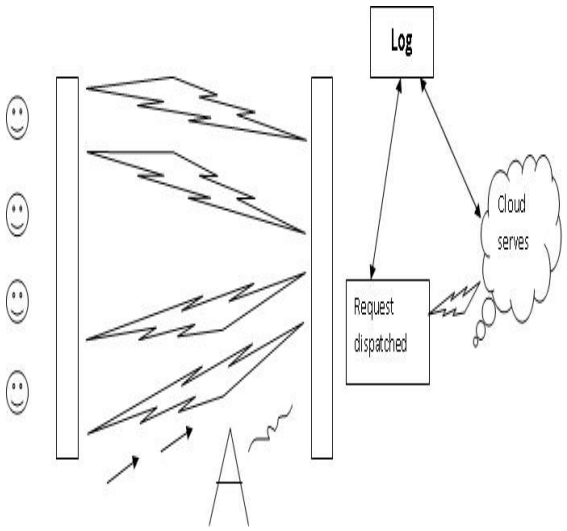
For each r_S in RS

RECEIVE (r,r_s)

DECR (r,k)

RS=r

End loop



CCFE Algorithm:

Encoding:

CCFE 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.

It is 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. CCFE is commonly used in a CCFE 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 CCFE takes 24bits in total. These 3 bytes are then converted into four printable characters from the ASCII standard.

The algorithm's name CCFE comes from the use of these 64 ASCII characters. The ASCII characters used for CCFE 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, CCFE 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 CCFE. If there was only one significant input byte, only the first two CCFE digits are picked (12 bits), and if there were two significant input bytes, the first three CCFE digits are picked (18 bits). '=' characters might be added to make the last block contain four CCFE characters.

Example:-

Text:	M	a	n
ASCII:	77	97	110
Bit pattern:	01001101	01100001	01101110
6bit split:	010011	010110	000101 101110
Index:	19	22	5 46
CCFE code:	T	W	F u

Padding:

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

Example1:

Input:
India is my country**.**
Output:
SW5kaWEgaXMgbXkgY291bnRyeQ==

Exapmple2:

Input:
India is a democratic country
Output:
SW5kaWEgaXMgYSBkZW1vY3JhdGljIGNvdW50cnk=

Algorithm3: My CCFE (encoding) algorithm:

Input → raw string

Output → CCFE encoded format

Step1: initialization

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

// 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 values ← all equivalent ASCII

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

Decoding:

When decoding CCFE 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**
My CCFE (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

Smart Router:

Initialization:

J_s←Java service

d_s←document service

G_s←groory service

$\sum C \leftarrow 0$ //request Categorization

n←0 //request count

$\sum R$ ←resoluter

$\sum AR$ ←all request vector

$\psi \leftarrow 8$ //threshold Value

$\sum S \leftarrow$ {java service, document service, groory service}.

For each r in A_R

Loop start

C←r

T_p Dup (c)

If ($F_P \epsilon Js | F_p \varepsilon d_s | t_p \epsilon G_s$)

Cat=s

R←REQEST2S (Fp,S)

ψ+=1

If ((ψ%8)==0)

ALERT(R)

End if

End for

Smart router Explanation:

In this algorithm we are using a centralized router where it takes all the requests and validates them. So that servers are given right accessibility from the clients that are requesting. The centralized router here is mentioned as LDAP service. There are 3 services here likely java service, document service and groovy service. The requests taken from the client are validated and routed to that particular service.

Conclusion:

In this project we are having a different approach when compared to remaining services. This is achieved by having an interface called LDAP. This LDAP will be acting like an interface which routes all the requested clients to their respective servers by validating them. We are even providing another feature like encryption which encrypts the data and sends them to the respected servers. Here the encryption is done for keys that have to be maintained in a separate log while forwarding the request to a particular server. Thus the servers and clients are matched perfectly

References:

[1] W.M. Zheng. Opportunities and Challenges to Cloud Computing.

[2]http://www.wsncs.zjut.edu.cn/download/20101204153234194.pdf, 2010.10.19

[3]M.V. Luis, R. M. Luis, C. Juan, L. Maik. A Break in the Clouds: Towards a Cloud Definition. Computer Communication Review, vol.39, pp.50-55, 2009

[4] http://en.wikipedia.org/wiki/E-learning

[5]http://www.vmware.com/company/news/releases/virtual_datacenter_os_vmworld08.html

[6] http://en.wikipedia.org/wiki/Data_center

[7]C Li, Z. H. Deng. On the Value of Cloud Computing. Library and Information,No4,pp,42-46,2009

[8] Li Jiahou. Cloud computing service in educational technology. Journal of Distance Education

[9]Yizeng Chen, Xingui Li, Fangning Chen Overview and Analysis of Cloud Computing research and application

[10]Hall Mark Everett . "SaaS Surprises" Computer World 2009,12:p19-22