International Journal of Engineering Research And Advanced Technology (IJERAT)
ISSN: 2454-6135 [Volume. 02 Issue.03, March-2016]

Efficient Searching Of Cloud Data Using Multi-String Matching Algorithm

Deepa M , Mrs. S. Vijayalakshmi

Department of CSE

IFET College of Engineering

Villupuram,

India

Abstract- *The cloud contains large volume of data that are stored by the cloud data owners. These cloud data will be encrypted by using the Symmetric Encryption Algorithm for data confidentiality. This makes searching of the documents/data difficult. Thus the efficient multi string matching algorithm called the Aho-Corasick algorithm is used. This algorithm gets the search text as the input keyword and provides the output file by matching it with the predefined keyword. This algorithm is used to find all occurrences of set of patterns given in the text string. The Aho-Corasick algorithm has the wide application over the text and pattern mining whereas the existing system uses the fuzzy keyword search.*

Keyword: *Encrypted data, Search, Aho-Corasick, Pattern mining***.**

I.INTRODUCTION

As we know, the cloud is the one that consists of large volume of data that are outsourced to the servers by the cloud data owners. Mining these cloud data will be difficult since they will be in the encrypted form. This is major problem for most of the cloud applications and to satisfy the user without providing any false data. For example if the user provide the input text string consists of two text string ,the search will be performed for that first keyword and again the search is performed for the other keyword. This makes the search more difficult and less efficient. Instead of making the keyword search, the pattern matching is introduced.

The efficient pattern matching algorithm is Aho-Corasick algorithm. This algorithm gets the input as the text string provided by the user. Also the algorithm forms the finite state machine with the predefined keyword. The predefined keyword will be provided by the cloud data owner while encrypting the file.

First of all, the cloud data owner will authorize the local server by using digital signature. This authorized server will have predefined set of keywords. By using this keyword, the local server will form the finite state machine which is used by the Aho-Corasick algorithm. In between to make the search efficient, the local server will form the index and provide the rank for the files. The ranking will be given based on the frequency of using the file by the particular user. Here the problem of supporting efficient ranked keyword search is solved for achieving effective utilization of remotely stored encrypted data in cloud computing. The proposed method supports both the multi-keyword query and support result ranking. List of keywords are provided only to the authorized server and also it was encrypted. This makes the privacy of the keyword be protected

2. RELATED WORK

There are many keyword search algorithms which enabled the keyword search easy. For example Jin Li, Qian Wang , Cong Wang, Ning Cao , Kui Ren , and Wenjing Lou explained the fuzzy keyword search over encrypted data in cloud computing. Fuzzy keyword search greatly enhances system usability by returning the matching files when users' searching inputs exactly match the predefined keywords or the closest possible matching files based on keyword similarity semantics, when exact match fails. But the fuzzy keyword searches only the keyword which makes the search less efficient.

Enabling secure and efficient ranked keyword is the paper which provides the concept of ranked search. Cloud computing economically enables the paradigm of data service outsourcing. However, to protect data privacy, sensitive cloud data have to be encrypted before outsourced to the commercial public cloud, which makes effective data utilization service a very challenging task. Although traditional searchable encryption techniques allow users to securely search over encrypted data through keywords, they support only Boolean search and are not yet sufficient to meet the effective data utilization need that is inherently demanded by large number of users and huge amount of data files in cloud. In this paper, we define and solve the problem of secure ranked keyword search over encrypted cloud data.

3. PROPOSED SYSTEM

The proposed system deals with the secure rank based multi keyword search algorithm called the Aho-Corasick algorithm. Aho-Corasick is found to be the efficient algorithm for multiple string matching that finds all occurrences of the pattern present in the files. The algorithm consists of two parts: The first part is building of the tree (trie) from keywords we want to search for. The second part is searching the test for the keywords using the previously built tree. The Aho-Corasick algorithm is the efficient algorithm that is used for pattern matching. The following explains the algorithm briefly. Aho-Corasick is found to be the efficient algorithm for multiple string matching that finds all occurrences of the pattern present in the files.

The algorithm consists of two parts:

1. The first part is building of the tree (trie) from keywords we want to search for.
2. The second part is searching the test for the keywords using the previously built tree.

The tree is a finite state machine, which is a deterministic model of behavior composed of a finite number of states and transitions between those states. In the first phase of tree building, keywords are added to the tree where the root node is just a place holder and contains links to other letters. The algorithm locates all occurrences of any of a finite number of keywords in a string of text.

Consists of two parts:

1. constructing a finite state pattern matching machine from the keywords
2. using the pattern matching machine to process the text string in a single pass.

Our problem is to locate and identify all substrings of x which are keywords in K.

1. K : K={$y_1, y_2, \ldots, y_k$} be a finite set of strings which we shall call keywords
2. x : x is an arbitrary string which we shall call the text string.

The behavior of the pattern matching machine is dictated by three functions: a goto function g, a failure function f, and an output function output.

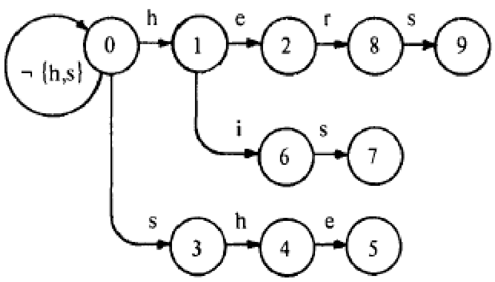
1. g *(s,a) = s' or fail* : maps a pair consisting of a state and an input symbol into a state or the message *fail*.
2. f *(s) = s'* : maps a state into a state, and is consulted whenever the goto function reports *fail*.
3. output *(s)* = *keywords* : associating a set of keyword (possibly empty) with every state.

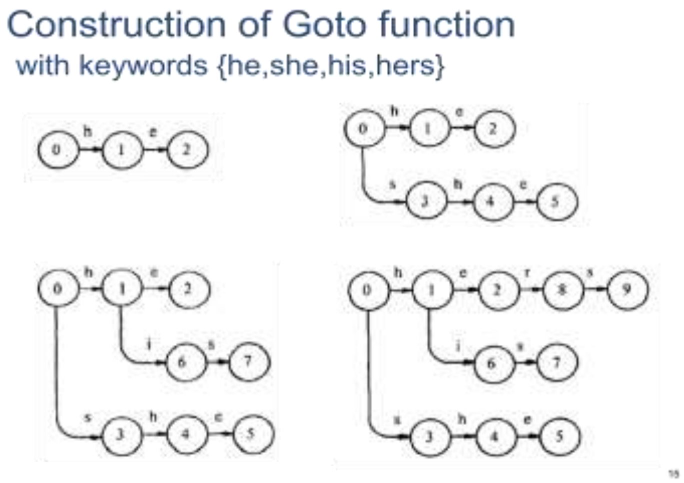
A. Example

Dictionary keywords {he,she,his,hers}

1. goto function

a. New vertices and edges to the graph, starting at the start state.
b. Add new edges only when necessary.
c. Add a loop from state 0 to state 0 on all input symbols other than the first one in each keyword.





2. *Failure function*

a. *Depth of s*: the length of the shortest path from the start state to state *s*.

b. The states of depth *d* can be determined from the states of depth d-1.

c. Make f(s)=0 for all states *s* of depth 1.

d. Compute failure function for the state of depth *d*, each state *r* of depth *d-1*. If g(*r*,*a*)=*fail* for all *a*, do nothing. Otherwise, for each a such that g(*r*,*a*)=*s*, do the following：

e. Set state=f(*r*).Execute state ←f(*state*) zero or more times, until a value for *state* is obtained such that g(*state*,*a*)≠*fail*. Set f(*s*)=g(*state*,*a*) .

B. Algorithms

Algorithm 1. Pattern matching machine.

Input. A text string x = $a_1\ a_2\ \ldots\ a_n$ where each a_i is an input symbol and a pattern matching machine M with goto function g, failure function f, and output function *output*, as described above.

Output. Locations at which keywords occur in x.

Method.

begin

state ← 0

for *i* ← 1 until n **do**

begin

while *g (state, a_i) = fail*

do *state ← f(state)*

state ← *g (state,* a_i *)*

if *output (state)≠ empty*

then

begin

print *i*

print *output (state)*

end

end

end

Algorithm 2. Construction of the goto function.

Input. Set of keywords $K = \{y_l, y_2, \ldots\ldots y_k\}$.

Output. Goto function g and a partially computed output function

output.

Method. We assume *output(s)* is empty when state s is first created, and *g(s, a) = fail* if a is undefined or if *g(s, a)* has not yet been defined. The procedure *enter(y)* inserts into the goto graph a path that spells out y.

begin

newstate ← *0*

for i ← 1 until k **do** *enter(*y_i *)*

for all *a* such that *g(0, a) = fail*

do *g(0, a)* ← *0*

end

procedure *enter(*$a_1 a_2 \ldots a_m$): **begin**

state ← 0; j ← 1

while *g (state,* a_j *)≠ fail* **do**

begin

state ← *g (state,* a_j*)*

j ← *j + l*

end

for p ← j until m **do**

begin

newstate ← *newstate + 1*

g (state, a_p *)* ← *newstate*

state ← *newstate*

end

output(state) ← *{* $a_1\ a_2 \dots a_m$*}*

end

Algorithm 3. Construction of the failure function.

Input. Goto function g and output function *output* from Algorithm 2.

Output. Failure function fand output function *output.*

Method.

begin

queue ← *empty*

for each a such that *g(0, a) = s≠0* **do**

begin

queue ← *queue* ∪*{s}*

f(s) ← *0*

end

while *queue ≠ empty* **do**

begin

let r be the next state in queue

queue ← *queue* - *{r}*

for each asuch that

g(r, a) = s≠fail

do

begin

queue ← *queue* ∪*{s}*

state ← *f(r)*

while *g (state, a) = fail*

do *state* ← *f(state)*

f(s) ← *g(state, a)*

output(s) ←*output(s)* ∪ *output(f(s))*

end

end

end

Eliminating failure function

Failure function Using in algorithm 1

δ(*s, a)*, a next move function δ such that for each state *s* and input symbol *a*.

By using the next move function δ, we can dispense with all failure transitions, and make exactly one state transition per input character.

Algorithm 4. Construction of a deterministic finite automaton.

Input. Goto function g from Algorithm 2 and failure function f from Algorithm 3.

Output. Next move function 8.

Method.

begin

queue ← *empty*

for each symbol *a* **do**

begin

δ(0, *a)* ← *g(0, a)*

if *g (0, a)* ≠ *0*

then *queue* ← *queue* ∪ *{g (0, a) }*

end

while *queue* ≠ *empty* **do**

begin

let r be the next state in *queue*

queue ← *queue* - {r}

for each symbol a

do

if g(r, *a) = s ≠ fail*

do

begin

queue ← *queue* ∪ {s}

δ(*r, a)* ← *s*

end

else δ(*r,a)* ←δ(*f(r), a)*

end

end

4. RESULTS

Attractive in large numbers of keywords, since all keywords can be simultaneously matched in one pass. Using Next move function can potentially reduce state transitions by 50%, but more memory. Spend most time in state 0 from which there are no failure transitions. This makes the search efficient when compared to the fuzzy based search. The below diagram shows the login page in figure1.

Figure1. Login page.

5. CONCLUSION

The proposed system consists of the four modules. The first module called the encryption module which is used to encrypt the user and the local server. The encryption module encrypts the user by using the any symmetric encryption algorithm. The second module consists the string matching module. The Aho-corasick algorithm is used in this module. The Aho-corasick algorithm is an efficient algorithm which is used to find the pattern efficiently. This can be extended by combining the Aho-corasick algorithm with the fuzzy keyword search which is the existing method. The combination of the both makes the search efficient.

REFERENCES

1. Song, D. Wagner, and A. Perrig.: "Practical Techniques for Searches on Encrypted Data." in Proc. of IEEE Symposium on Security and Privacy‟ (2000).
2. D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano.: "Public Key Encryption with Keyword Search." in Proc. of EUROCRYP‟04, volume 3027 of LNCS. Springer (2004).
3. Y.-C. Chang and M. Mitzenmacher.: "Privacy Preserving Keyword Searches on Remote Encrypted Data." in Proc. of ACNS‟05 (2005).
4. J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou.: "Fuzzy Keyword Search over Encrypted Data in Cloud Computing." in Proc. of IEEE INFOCOM‟10 MiniConference (2010).
5. Wang, N. Cao, K. Ren, and W. Lou.: "Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data." IEEE Transactions on parallel and distributed systems, vol. 23,no. 8 (2012).
6. C. Wang, N. Cao, J. Li, K. Ren, and W. Lou.: "Secure Ranked Keyword Search over Encrypted Cloud Data." in Proc. of ICDCS‟10 (2010).
7. R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky.: "Searchable Symmetric Encryption: Improved Definitions and Efficient Construction." in Proc. of ACM CCS‟06 (2006).
8. Remya Rajan.: "Efficient and Privacy Preserving Multi User Keyword Search for Cloud Storage Services." International Journal of Advanced Technology And Engineering Research (IJATER), ISSN 2250 - 3536,Vol 2,Issue 4 (2012).
9. Zeeshan Ahmed Khan, R.K Pateriya.: "Multiple Pattern String Matching Methodologies" A Comparative Analysis (2012).
10. H. Witten, A. Moffat, and T. C. Bell.: "Managing Gigabytes: Compressing and Indexing Documents and Images." Morgan (1994)