Preserving security using crisscross AES and FCFS scheduling in cloud computing

Article in International Journal of Advanced Intelligence Paradigms - January 2019
DOI: 10.1504/IJAIP.2019.10017743

2 authors, including:

K. Ramkumar
Kings Engineering College, Chennai, India

6 PUBLICATIONS 0 CITATIONS

All content following this page was uploaded by K. Ramkumar on 05 January 2019.
The user has requested enhancement of the downloaded file.
Preserving security using crisscross AES and FCFS scheduling in cloud computing

K. Ramkumar*

Department of Computer Science and Engineering,
Manonmaniam Sundaranar University,
Thirunelveli, Tamilnadu, India
Email: ramkumar.k.2875@gmail.com
*Corresponding author

G. Gunasekaran

Meenakshi College of Engineering,
West K.K Nagar, Chennai, Tamilnadu, India
Email: Gunasekaran_Dr@rediffmail.com

Abstract: Cloud computing is a developing technology in distributed computing which provides pay service model as per user need and requirement. Cloud includes the collection of virtual machine which have both computational and storage facility. The objective of cloud computing is to provide effective process to hyper distributed resources. Nowadays, cloud system is developing and faces many challenges, two of them are scheduling process and other main challenge is security. Scheduling states about how scheduler adapts its scheduling strategy, according to the changing set of morals to control the order of work to be implemented by a computer system. In this research paper, a scheduling algorithm of collocate first come first server (FCFS) of supremacy elements is proposed where the system efficiency is improved using FCFS in parallel manner. FCFS is simple and fast. To address security problem, crisscross advance encryption standard (AES) is proposed by increasing the security in the cloud through the grid manner. AES uses an identical key for both encrypting and decrypting the text. Aggregate of this proposed work is to enhance the system efficiency and security by using the both crisscross AES and collocate FCFS of supremacy elements.

Keywords: cloud computing; first come first server; FCFS; advance encryption standard; AES; security.

Reference to this paper should be made as follows: Ramkumar, K. and Gunasekaran, G. (2019) ‘Preserving security using crisscross AES and FCFS scheduling in cloud computing’, Int. J. Advanced Intelligence Paradigms, Vol. 12, Nos. 1/2, pp.77–85.

Biographical notes: K. Ramkumar has more than 15 years of rich academic experience including IT. Presently he is a PhD scholar in CSE Department at Manonmaniam Sundaranar University, Tirunelveli, Tamilnadu, India. Prior to this assignment, he has worked in various engineering colleges in Chennai Tamilnadu India. He worked as Associate Consultant in Satyam Computer Services Ltd and Project Consultant in CGI. He has already published four research papers in four reputed international journals and conferences. He also presented a paper in security issues in cloud computing in an international conference in Bangkok, Thailand.
G. Gunasekaran has more than 20 years of rich academic experience and is presently working as a Principal and a Professor in Computer Science and Engineering Department of Meenakshi College of Engineering, Chennai. Prior to this assignment, he was working in SRM Easwari Engineering College, Panimalar Engineering College Engineering Colleges. He has published research papers in six reputed international journals and conferences. He completed his PhD in Jadavpur University, Calcutta.

1 Introduction

Cloud computing is one of the developing technologies in distributed computing which provides pay service model as per user need and requirements. Cloud includes the collection of virtual machine which have both computational and storage facility. Main challenges in the cloud are that it permits services to be used without any understanding of their arrangement. Cloud computing works using economies of scale. It decreases the expenditure for start up companies, as they no need to buy their own software. Cloud services are issued ‘as a service’ over the internet, naturally in the form of platform as a service (PaaS), infrastructure as a service (IaaS), or software as a service (SaaS).

Scheduling (Kaur and Kinger 2014) allocates tasks to available resources based on the tasks’ qualities and need. The aim of scheduling is to enhance resource utilisation without lowering the quality of service (QOS) provided. The scheduling is based on either the resource or the jobs. In resource scheduling, the focus is on fairness of resource allocation whereas in job scheduling the focus is on the job allocation. There are two types of scheduling, i.e., resource scheduling and job scheduling. Resource utilisation refers to the degree of the resources utilised. Energy consumed is the amount of energy used by the resources of system. A good scheduling algorithm must provide maximum resource utilisation and also minimise energy consumption. The three stages of scheduling in cloud are: resource discovering, resource selection and task allocation.

Cloud computing security issues in IaaS (Arora et al., 2012) declares cloud computing as current catchword in market. In Suresh and Prasad (2012), various security issues and security algorithms encountered are discussed. In Li et al. (2015), dissimilar effects on one-to-many order preserving encryption (OPE) is discussed and new algorithm was proposed. While increasing utilisation of resources with QOS (Shakkeera et al., 2013) efficient load balancing algorithm was proposed. Kliazovich et al. (2016) proposed communication aware-directed acyclic graphs (CA-DAG) model of cloud computing applications. The proposed CA-DAG model optimises resource allocation and helps in developing novel scheduling schemes of improved efficiency. Advanced encryption standard (AES) (Selent, 2010) developed by two Belgian cryptographers, Vincent Rijmen and Joan Diemen and implemented using a combination of XOR operations, quartet substitution with S-box, rotations between row, columns and a mix column.

This paper presents two important algorithms for improving security and scheduling in cloud computing where security issue rectified by introducing crisscross AES. Parallel implementation of AES (Pachori et al., 2012) is done to improve efficiency and security to the cloud system. The paper is organised as follows: Section 2 discusses on collocate first come first server (FCFS) of supremacy elements, Section 3 discusses on the proposed crisscross AES algorithm, Section 4 provides the experimental results and Section 5 provides the conclusions.
2 Collocate FCFS of supremacy elements

The aim of cloud task scheduling is to maintain high system performance and allocate many handled resources to applications. The complexity of scheduling problem gears with the size of the task and becomes highly complex to solve efficiently (Shimpy and Sidhu, 2014). In its performance study, the tasks arrive in random and the maximum time a job can wait for service is noted. In M/M/1 systems using FCFS or advanced FCFS scheduling algorithm, clear terms for incomplete work allocation, job loss ratio, and CPU intake for M/M/1 + M systems are emulative. In M/M/1 + M, the last M indicates that exponentially allocation of job leniency. This approach produces a moderate level of time consuming and less efficiency when included with advanced FCFS. But in our proposed efficient ideal approach for security and scheduling (EIASS) in cloud computing, we have implemented a novel concept of collocate FCFS of supremacy elements that improves the system performance and reduce time consumption through the collocate elements.

Figure 1 Algorithm for scheduling using collocate FCFS of supremacy elements

Input: n number of inputs sends through parallel manner
Output: scheduled data

Assume:
Burst time (BT) = 30;
Waiting time (WT) = 30;
Turnaround time (TAT) = 30;
Average waiting time (AWT) = 0;
Average turnaround time (ATAT) = 0;
Add A;
For all i = 0, i < n < i++
  BT = i;
End for
WT [i] = 0;
For all i = 1, i < n < i++
  WT [i] = 0;
For all j = 0, j < I < j++
  WT[i] = BT [j]
End for
For all i = 0, i < n < i++
  TAT [i] = BT [i] + WT [i];
  AWT = WT [i];
  ATAT = TAT [i];
End for
// implemented with collocate (parallel) of supremacy elements
A1(int * AWT, int * ATAT, int * Z)
Figure 1  Algorithm for scheduling using collocate FCFS of supremacy elements (continued)

Assume:
int i, p, int Y = i, min data [i] = p;
while (Y)
int S = (Y-1)/2
if (min data [s] <= min data [Y])
interchange (S, Y); Y = S; End while
collect (y)
for all int i = 0, i < min count, i++
return min data [i];
End for
End

A2 (int *AWT, int * ATAT, int * Z)
Assume:
Int i, p, int Y = i, min data [i] = p;
While (Y)
int S = (Y-1)/2
if (min data [s] <= min data [Y])
interchange (S, Y);
Y = S;
End while
collect (y)
for all int i = 0, i < min count, i++
return min data [i];
End for
End
Call A2
Wait;
End

Assume:
Parallel process = M; // end of collocate process
M = A1 + A2
End

End

The steps of the proposed algorithm are:
1  start process
2  read number of processes as a parallel process
3  read burst time
4  read arrival time
3 Proposed crisscross AES algorithm

In the proposed algorithm, improved AES is implemented in a grid manner. It has four nodes, sub byte, shift row, mix column and add round key. Sub key indicates non-Feistel cipher that can code and decode blocks of 128. Shift row offsets bytes in each row as instructed in algorithm. The mix column process is essentially matrix multiplication. Add round key transformation is deployed via simply combining state and key modulo 2 within the sub processor. Proposed crisscross approach AES is composed of ten rounds at four levels. First level includes three times of M-1 and second round has three times of M-1 vice versa till the third level. Fourth round has combination of all three levels which is forwarded to the server. This process is performed via crisscross approach that reduces time consumption and increase the system efficiency. The algorithm for crisscross AES algorithm is presented in Figure 2.

Figure 2 Algorithm for crisscross AES

Input: plain text of user input
Output: cipher text
Create (R0, R1, ………Rn-1)
for all Ri where 0 < i < n-1
Si = Plain text [(i * 128), (i * 128) + 1, (i * 128) + 2………..(i * 128) + 127]  
Call AES (Si, l)
End for

Proceed AES in a grid manner (Alg. 2)
Algorithm 2: AES with grid manner (Crisscross)
Input: plain text
Output: Encrypted data or cipher text
//Crisscross AES

Initiate:
Create (R0, R1, R2)
S0 = Plain text [0, 31] // first three rounds M-1 times
S1 = Plain text [31, 63] // second three rounds M-1 times
S3 = Plain text [63, 127] // third three rounds M-1 times
For all Ri, 0 <= j <= 1
For j = 0 to 9
   Call operation1 on Si
Figure 2  Algorithm for crisscross AES (continued)

Wait;
Call operation 2 on Si
Wait;
Call operation on Si
End for
Call last round [S0 + S1 + S2] // Last round (combination of all)
End for
End

Figure 3  Crisscross AES structure

The structure of crisscross AES shown in Figure 3 includes server, four nodes and finally cipher text unit. Here, it consists of following procedures:

1. First the plain text can be forwarded to the node 1, node 2, and node 3.
2. The plain text in the size of 128 bit and node 1 has process with three rounds, node 2 has process with three rounds, and node 3 has process with three rounds.
3 Here each and every three rounds have units like, subtype, shift row, mix column and add round key.

4 Each and every node has performed its operation at M-1 times and every processed output send to the final node. Last node has performed combine work of all rounds.

5 Finally the combined output of cipher text will send to the server and the server forward it to the user.

4 Proposed architecture

Figure 4 shows the implementation of EIASS in cloud computing, where user accesses the both IaaS and SaaS services in cloud. In general, cloud provides lack of scheduling and security to user. To avoid this problem, a new concept of crisscross AES for security measures and collocate FCFS of supremacy element for scheduling process is implemented.

Figure 4  Implementation of EIASS in cloud computing (see online version for colours)

5 Results and discussion

The performance analysis of FCFS is shown in Table 1. Here average time (AT) depends on number of processes, computation time (CT) is only seconds of setup time; waiting time (WT) is the waiting period in queue and turn around time (TAT) is sum of periods spent waiting to get into memory, waiting in ready queue, executing and doing input output. Finally, this algorithm has produce WT as 43.
### Table 1  Performance analysis of FCFS

<table>
<thead>
<tr>
<th>Process</th>
<th>AT</th>
<th>BT</th>
<th>CT</th>
<th>TAT</th>
<th>WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P2</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>P3</td>
<td>2</td>
<td>5</td>
<td>11</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>P4</td>
<td>1</td>
<td>3</td>
<td>14</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>P5</td>
<td>3</td>
<td>4</td>
<td>18</td>
<td>15</td>
<td>11</td>
</tr>
<tr>
<td>P6</td>
<td>2</td>
<td>2</td>
<td>20</td>
<td>18</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 2 presents the scheduling process in the priority manner. Here, it provides WT as 52.

Table 3 explains presents performance analysis of collocate FCFS in parallel way with priority and produce WT as 36 which is the least WT. This will increase the system performance and reduce the overall time consumption.

### Table 2  Performance analysis of priority queue

<table>
<thead>
<tr>
<th>Process</th>
<th>Priority</th>
<th>BT</th>
<th>WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>P2</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>P3</td>
<td>5</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>P4</td>
<td>3</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>P5</td>
<td>6</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>P6</td>
<td>2</td>
<td>2</td>
<td>13</td>
</tr>
</tbody>
</table>

### Table 3  Performance analysis of collocate FCFS of supremacy elements

<table>
<thead>
<tr>
<th>Process</th>
<th>AT</th>
<th>BT</th>
<th>CT</th>
<th>TAT</th>
<th>Priority</th>
<th>WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P2</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>P3</td>
<td>2</td>
<td>5</td>
<td>11</td>
<td>9</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>P4</td>
<td>1</td>
<td>3</td>
<td>14</td>
<td>13</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>P5</td>
<td>3</td>
<td>4</td>
<td>18</td>
<td>15</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>P6</td>
<td>2</td>
<td>2</td>
<td>20</td>
<td>18</td>
<td>2</td>
<td>12</td>
</tr>
</tbody>
</table>

### Table 4  Analysis between AES and crisscross AES

<table>
<thead>
<tr>
<th>Input size</th>
<th>AES</th>
<th>Crisscross AES</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>56</td>
<td>64</td>
</tr>
<tr>
<td>108</td>
<td>40</td>
<td>57</td>
</tr>
<tr>
<td>246</td>
<td>110</td>
<td>75</td>
</tr>
<tr>
<td>320</td>
<td>162</td>
<td>147</td>
</tr>
<tr>
<td>781</td>
<td>165</td>
<td>152</td>
</tr>
<tr>
<td>900</td>
<td>260</td>
<td>172</td>
</tr>
<tr>
<td>695</td>
<td>212</td>
<td>144</td>
</tr>
<tr>
<td>Average</td>
<td>3.08</td>
<td>3.82</td>
</tr>
</tbody>
</table>

From Table 4, analysis between AES and crisscross algorithms is determined for different input size. Finally, the crisscross AES provides high throughput than the AES. The average throughput can be calculated by using:

$$\text{Average throughput} = \frac{\text{Total plain text}}{\text{Total Encryption (ENC) or Decryption (DNC)}}$$

### 6 Conclusions

We have proposed a scheduling algorithm using collocate FCFS of supremacy elements for improving the privacy and security in cloud computing. To address security problem, crisscross AES algorithm is proposed which increases the security in cloud through grid. This implementation has produced high security and time reduction through the collocated FCFS of supremacy elements. In future work, the effect of increased complexity of the proposed algorithms on throughput and WT is planned to be analysed. Future work includes investigating the efficacy of other cryptography algorithm and also to implement parallel computing to increase their performance.

### References