Researches on Grid Security Authentication Algorithm in Cloud Computing

Keshou Wu *
Department of Computer Science and Technology, Xiamen University of Technology, Xiamen 361005, China
Email: kollzok@yahoo.com.cn

Lizhao Liu
Department of Computer Science and Technology, Xiamen University of Technology, Xiamen 361005, China
Email: 493107149@qq.com

Jian Liu
College of Information Sciences and Technology, The Pennsylvania State University, PA, USA
Email: 33095944@qq.com

Weifeng Li, Gang Xie, Xiaona Tong and Yun Lin
KOLLZOK Intelligent Technology Co., Ltd, Xiamen, 361024, China
Email: kollzok@yahoo.com

Abstract—Focusing on multi-machine distributed computing security problems in cloud computing, the paper has proposed a grid distributed parallel authentication model based on trusted computing, which can realize simultaneous verification of grid authentication and grid behavior on upper layer of SSL and TLS protocols. Adaptive grid authentication method is established applying adaptive stream cipher framework; an adaptive stream cipher heuristic code generator and k-means heuristic behavior trust query function is proposed and acted as authentication kernel. Through comparison of the test results of TLS and SSL authentication protocol and the new grid authentication method, the effectiveness of the new grid authentication method has been explained.

Index Terms—distributed computing; trusted computing; cloud computing; grid behavior; grid authentication; TLS; SSL

I. INTRODUCTION

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that provide communications security over the Internet[1][2]. TLS and SSL encrypt the segments of network connections above the Transport Layer, using symmetric cryptography for privacy and a keyed message authentication code for message reliability. Several versions of the protocols are in widespread use in applications such as web browsing, electronic mail[3][4], Internet faxing, instant messaging and voice-over-IP (VoIP). TLS is an IETF standards track protocol, last updated in RFC 5246 and is based on the earlier SSL specifications developed by Netscape Corporation[5][6][7]. The TLS protocol allows client/server applications to communicate across a network in a way designed to prevent eavesdropping and tampering. A TLS client and server negotiate a stateful connection by using a handshaking procedure. During this handshake, the client and server agree on various parameters used to establish the connection's security[8][9][10].

Cloud computing refers to the provision of computational resources on demand via a computer network. In the traditional model of computing, both data and software are fully contained on the user's computer; in cloud computing, the user's computer may contain almost no software or data (perhaps a minimal operating system and web browser only), serving as little more than a display terminal for processes occurring on a network of computers far away[11][12]. A common shorthand for a provider's cloud computing service (or even an aggregation of all existing cloud services) is "The Cloud". The most common analogy to explain cloud computing is that of public utilities such as electricity, gas, and water. Just as centralized and standardized utilities free individuals from the vagaries of generating their own electricity or pumping their own water, cloud computing frees the user from having to deal with the physical, hardware aspects of a computer or the more mundane software maintenance tasks of possessing a physical computer in their home or office. Instead they use a share of a vast network of computers, reaping economies of scale [13][14].

Grid computing is a term referring to the combination of computer resources from multiple administrative domains to reach a common goal. The grid can be thought of as a distributed system with non-interactive workloads that involve a large number of files. What
distinguishes grid computing from conventional high performance computing systems such as cluster computing is that grids tend to be more loosely coupled, heterogeneous, and geographically dispersed. Although a grid can be dedicated to a specialized application, it is more common that a single grid will be used for a variety of different purposes. Grids are often constructed with the aid of general-purpose grid software libraries known as middle ware[15][16].

Trusted Computing (TC) is a technology developed and promoted by the Trusted Computing Group. The term is taken from the field of trusted systems and has a specialized meaning. With Trusted Computing, the computer will consistently behave in expected ways, and those behaviors will be enforced by hardware and software. In practice, Trusted Computing uses cryptography to help enforce a selected behavior. The main functionality of TC is to allow someone else to verify that only authorized code runs on a system. This authorization covers initial booting and kernel and may also cover applications and various scripts. Just by itself TC does not protect against attacks that exploit security vulnerabilities introduced by programming bugs[17][18].

II. GRID DISTRIBUTED PARALLEL AUTHENTICATION MODEL

If the grid entity A want to intact with grid entity B in a cloud, the grid entity A will first go into grid entities heuristic trusted query, this process need to calculate trusted value in grid domain and out of grid domain; at the other hand it need to compute the grid entity adaptive authentication. If the verify behavior reach its gate value the information will be sent to the decision module besides the information of grid entity adaptive authentication, then the decision module will give the comprehensive information of the trusted value of grid B for A. During the process grid B will interact with grid entity adaptive authentication module to give sufficient information or else it will be rejected.

III. ADAPTIVE GRID AUTHENTICATION VERIFY FRAME

Adaptive grid authentication verifies can realize signal self- detection and self-adjusting. [19] The adaptive generator initialization within the production of continuous or intermittent output with automatic recognition and adjustment function of the generate signal, through the design of reference models or self-tuning controller module can be achieved on the output or received signal real-time adjustment and dynamic match. Adaptive encryption control principle is as Figure 2. First initiative the clock module and the clock stimulus module as a self-reference model, since the self-reference model will reconstruct when the detective signal received from the self-detection module does not match, and the reconstructed reference model is not dependent on external stimulation, which depends only on the initial algorithm $T$. This means that as long as both encryption and decryption have the same reference model, after the same initialization, they can always get synchronous control signal. For example, the use of the two CMOS unit can keep output synchronism at $810 / kS$. Take the output signal from the self-reference model as the first stage parameter of chaos cascade module, the output signal of the first stage of chaotic module as the input.
signal of self-tuning module and at the same time, as the input signal of the second stage Logistic generator, the output signal of the second stage Logistic generator as the input signal of the third stage or input signal of key generator; the output signal of the third stage and key k together as the initial key of key generator. The self-detection module consists of two detectors, which are responsible for adding state value \( w \) to the plaintext and testing encrypted state value \( w' \) and channel feedback state value \( w'' \). When the encrypted value and the channel feedback state values are abnormal, feedback should be done to the reference model, then the self-reference model will update with a new reference model according to the current state value of initial algorithm, thus to update all the output signals of chaotic module, key generators will also update new key stream without changing of key k. The receiver uses the decoder to return \( w'' \) to the sender first to confirm the acceptance of the cipher, at the same time offers \( w'' \) evaluation to the sender for the channel safety index testing thus to determine whether there is need to resend the cipher or renew the reference model.

**Figure 2. Adaptive grid authentication method**

**IV. HEURISTIC CODE GENERATOR AND HEURISTIC BEHAVIOR TRUST QUERY FUNCTION**

**A. Structure of heuristic code generator**

Supposed that the grid entity have some shape volume and weight[20], the length of grid entity is L, the width W and the height H, the direction of the layers vary from each other due to different position of grid entity, and the x, y, z mentioned above correspondingly represents different value[21]. If the grid entity is located behind of grid domain, then x equals to W, y equals to H, and z equals to L, which means the direction of the layer is along with the length of entity. If the grid entity is located in the side of compartment, then x equals to L, y equals to H, and z equals to W, and the direction of the layer is along with the width of compartment. If there are two entities set respectively behind and side of the grid, then x equals to W, y equals to L, z equals to H, and the direction of the layer is along with the height of grid. Call the best layer loading program and the plane optimal layout program to solve these and the specific steps are as follows:

1. Enter the grid entity size L, W, H, the grid box size \( s_1, s_2, \) and \( s_3 \) (make \( s_1 \geq s_2 \geq s_3 \)). Assign x, y, z with L, W, H according to the position of the grid entity;
2. Call the best layer loading program to compute \( a, b, c \);
3. Calculate the optimal layout of each layer:
   1. If \( a \neq 0 \), make \( s_m = s_2, \ s_n = s_3 \), call the plane optimal layout program generator \((s_2, s_3)\);
   2. If \( b \neq 0 \), make \( s_m = s_1, \ s_n = s_3 \), call the plane optimal layout program generator \((s_1, s_3)\);
   3. If \( c \neq 0 \), make \( s_m = s_1, \ s_n = s_2 \), call the plane optimal layout program generator \((s_1, s_2)\);
4. Calculate the total number of the loading container
   \[ \text{Sum} = a \text{genenator}(s_2, s_3) + b \text{genenator}(s_1, s_3) + c \text{genenator}(s_1, s_2) \]
5. Output \( \text{Sum} \), \( a, b, c \), and the corresponding values of the parameters of the layout.

Given a grid grid entity set \( C_0 = \{1,2,\ldots,n\} \), the set \( C_0 = \{0\} \) denotes the grid; \( d_x \) is the trusted value between
arbitrary two nodes \( i, j \) in \( C \) and \( C_0 \); \( q_i \) \((i=1, \ldots, n)\) is the demand of the grid \( i \); \( w \) is the maximum of the trusted capability of \( R \) is the number of the grid entity that needs to finish the verity, which is

\[
R = \left\lfloor \sum_{i=1}^{n} q_i / w \right\rfloor,
\]

(1)

“\( \left\lfloor \cdot \right\rfloor \)” is the rounded up function, such as \( \left\lfloor 6.2 \right\rfloor = 7 \); \( x_{ij}^r (r=1, \ldots, R, i \text{ and } j=0, \ldots, n, \text{ and where } i \text{ not equals to } j) \) is the decision variables, \( x_{ij}^r = 1 \) if and only if the \( r \) routine pass the arc \((i, j)\), otherwise \( x_{ij}^r = 0 \); \( y_i^r ((r=1, \ldots, R, i=1, \ldots, n) \) is the demand of the \( i \) grid which meets by the \( r \) routine; \( S' \) denotes the grid set served by the \( r \) routine, \( |S'| \) denotes the number of grid included in \( S \). There are some assumptions of the model:

(1) the trusted values between two nodes is symmetric, \( d_y = d_y' \);

(2) the trusted values of the nodes satisfy the triangular inequality, which is \( d_y + d_y' > d_y' \);

(3) all the grid entity start from the grid and back to grid after each delivery;

(4) every grid’s needs must be satisfied and can be done by one or more grid entity.

The objective of this problem is to arrange the routine to minimize the cost of delivery. The cost is represented by the total travelling trusted value. As the description above, the problem can be modeled as:

\[
\min \sum_{r=1}^{n} \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} x_{ij}^r
\]

(2)

\[
\sum_{j=0}^{n} x_{ij}^r = \sum_{j=0}^{n} x_{ij'}^0 \quad k = 0, \ldots, n; r = 1, \ldots, R
\]

(3)

\[
\sum_{i=0}^{n} x_{ij}^r \geq 1 \quad j = 0, \ldots, n
\]

(4)

\[
\sum_{i=0}^{n} \sum_{j=0}^{n} y_{ij} = q_i \quad i = 1, \ldots, n
\]

(5)

\[
\sum_{i=0}^{n} y_{ij} \leq w \quad r = 1, \ldots, R
\]

(6)

\[
\sum_{j=0}^{n} x_{ij}^r y_{ij} \geq y_{ij} \quad r = 1, \ldots, R; i = 1, \ldots, n
\]

(7)

\[
x_{ij}^r \in \{0, 1\} \quad i = 1, \ldots, n; j = 1, \ldots, n
\]

(8)

\[
q_i \geq y_{ij} \geq 0 \quad i = 1, \ldots, n; r = 1, \ldots, R
\]

(9)

The constraint (2) is to minimize the total travelling trusted value; constraint (3) means the flow conservation, that is, the number of grid entity is equal between entering and exiting of a node; Constraint (4) and (5) ensure that each node is visited at least one time and the requirement is satisfied; (6) shows that the edges between served grid \( s \) equals to the number of served grid \( s \) minus 1 in each route, (7) shows the trusted capability of grid; (8) shows that the grid is served only the grid pass.

Compute 1: input \( x, y, s_m, s_n \)

Compute 2: determine whether the grid \( A \) and \( B \) can be trusted in its domain, return 0 if not.

Compute 3:

\[
\{ y_i = \left\lfloor y_i / 2 s_n \right\rfloor; y_i \leq \left\lfloor y_i / s_m \right\rfloor; y_i + + \}
\]

\[
\{ \text{for } (x_i = \left\lfloor x_i / 2 s_m \right\rfloor); x_i \leq \left\lfloor x_i / s_m \right\rfloor; x_i + + \}
\]

\[
\text{initialize } x_2 = y_2 = 0 \text{ and compute } x_2, y_4
\]

using

\[
\sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} x_{ij}
\]

(10)

\[
\sum_{i=0}^{n} x_{ij}^r y_{ij} \geq y_{ij} \quad r = 1, \ldots, R; i = 1, \ldots, n
\]

\[
\text{if } (s_n y_i / s_m) \geq \left\lfloor y_i / s_m \right\rfloor
\]

\[
\{ y_2 = \left\lfloor y_i / s_m \right\rfloor; x_3 = y_3 = 0 ; \text{Compute } x_4
\]

\[
\sum_{i=0}^{n} x_{ij} = |S| - 1 \quad r = 1, \ldots, R ; S \subseteq C - \{0\}
\]

\[
\text{if } (s_n x_2 - s_m (s_n x_2 / s_m) > s_m / 2)
\]

\[
\text{compute } x_3, x_4 \text{ using } q_i \geq y_{ij} \geq 0 \quad i = 1, \ldots, n; r = 1, \ldots, R
\]

\[
\text{else compute } y_2
\]

\[
\sum_{i=0}^{n} y_{ij} \leq w \quad r = 1, \ldots, R \quad \sum_{i=0}^{n} y_{ij} = q_i \quad i = 1, \ldots, n
\]

\[
\text{if } (x - s_m x_3 - s_n x_4 > s_n) \text{ compute } x_5, y_2 \text{ using } q_i \geq y_{ij} \geq 0 \quad i = 1, \ldots, n; r = 1, \ldots, R
\]

\[
\text{compute the total number of generator: } x_1 y_1 + x_2 y_2 + x_3 y_3 + x_4 y_4 + x_5 y_5
\]

\[
\text{compare and record the generator number and the agreement method.}
\]

\[
\text{Output the optimized result.}
\]

We can also

\[
\text{have } x_3 = \left\lfloor (x - s_m x_3 - s_n x_4) / s_n \right\rfloor
\]

\[
\text{y}_3 = \left\lfloor (y - s_m y_2) / s_m \right\rfloor
\]

\[
\text{The objective function is generator: } x_1 y_1 + x_2 y_2 + x_3 y_3 + x_4 y_4 + x_5 y_5
\]

\[
\text{When } s_n x_2 - s_m (s_n x_2 / s_m) > s_m / 2,
\]
\[ x_3 = \left[ s_p x_2 / s_m \right] + 1 \quad x_4 = \left[ \frac{(x - s_m x_3)}{s_n} \right]. \]

The objective function is generator = \( x_1, y_1 + x_2, y_2 + x_3, y_3 + x_4, y_4 \).

So in such verify model, the parameters can be solved as long as \( x_1 \) and \( y_1 \) are known. The range of \( x_1, y_1 \) are \( 0 \leq x_1 \leq x/s_m \) and \( 0 \leq y_1 \leq y/s_m \), which becomes to \( x_2 = x/2s_m \leq x \leq x/s_m \) and \( y_2 = y/2s_m \leq y \leq y/s_m \) on considering the symmetry of verify model 4. The objective optimization value can be found after the traversal of all the combinations of \( x_1, y_1 \).

B. k-means heuristic behavior trust query function

In many practical applications, the k-means clustering algorithm (k-means algorithm) which is based on partition clustering has been proven to be effective and generate good results[22][23]. The steps of a general k-means algorithm are:

1. Select k couples of initial cluster center;
2. Assign sample x which need to be classified to some cluster center one by one according to the minimum trusted value principle;
3. Calculate new value of every cluster center. Generally the new cluster center is the mean vector of the sample contained in the cluster field. The mean vector of the sample in k couples of cluster need to be calculated respectively.
4. Reclassify the sample and repeat iteration. The algorithm converges when every cluster center no longer moves, then calculation finishes.

The principle of k-means algorithm is to find k couples of partition with a least square error and make the generated result as compact and separate as possible. The k-means algorithm is relatively scalable and efficient dealing with large data sets and the complexity is \( O(nkt) \), in which n means the number of objects, k is the number of cluster, and t is the number of iterations. The case mainly discussed in this paper is that the demand of grid may be greater than the maximum trusted capacity of grid entity. Hence, it is prior to meet each grid wholly, and then merge the remaining part to other grid to meet.

Next the principle discussed is used to cluster the grid entities and determine the grid s served by the same grid. However, the SDVRP is a constraint clustering problem, the calculation may not converge, so the number of iterations \( N \) needs to be set to terminate forcibly and set the clustering evaluation criteria to select better clustering results. The clustering evaluation function used in this paper is:

\[
\text{Min}(\text{sumD}) = \sum_{j=1}^{g} \sum_{i \in C_j} d_{ij},
\]

\( C_i \) represents cluster j. The formula above calculates the sum of trusted value between every grid entity and the center in the cluster. Select the minimum sum as the best clustering result. The concrete steps are below:

Step 1: Find the grid entity whose demand is greater than or equal to the trusted capacity of grid. Split the demand \( q_i \) to two parts \( q^i_1 \) and \( q^i_2 \), and

\[
q^i_1 = w \left[ q_i / w \right],
q^i_2 = q_i - w \left[ q_i / w \right]
\]

“[ ]” means to round down, for example \([6.6] = 6\).

The demand of \( q^i_1 \) is individually met and the remained demand \( q^i_2 \) and the other entity are merged to some other circuit to meet. Modify the demand of the grid i to be \( q^i \);

Step 2: Randomly select R couples of initial cluster center \( 1^l, \cdots, R^l \) from the grid set \( C = 1, 2, \cdots, n \), and mark as set \( P^1 = 1^l, \cdots, R^l \). Initialize every cluster set \( C_i = \Phi(i = 1, \cdots, R) \), and set the value of the maximum number of iterations \( N \);

Step 3: Cluster the grid s. Calculate the trusted value \( d_{ij} \) between every grid entity and every cluster center, and find the nearest cluster center of every grid entity. The nearer the trusted value is, the higher priority the grid entity has to join the center. If the cluster wanted to join is full loaded, then choose the second nearest. When there is still remaining demand in the cluster, and if the adding of the demand make the total demand of cluster \( C_j \) exceed \( W \ (QC_j > W) \), compute the unmet demand of grid i, which is denoted by S, and transmit the unmet demand to other grid of \( C_i \). The transmission principle is: firstly find the grid entity (include grid i) whose demand is not less than \( S \) in cluster \( C_j \), then find the cluster whose residual demand \( SuQ_z = W - QC_z \geq S(z \in P^l - j) \). Compute the trusted value between these grid entities and these clusters and choose the grid entity with smallest trusted value to split. Guess the grid entity k and its corresponding cluster center \( p \), add k to cluster \( C_p \) and the unmet demand \( S \) is met by this route. If the residual demand of all clusters \( SuQ_z < S \), then select the cluster with largest residual demand to join until \( S \) is fully met. Repeat this step until all the grid s’ demands are met.

Step 4: Calculate the sum of the trusted value between every clustering grid entity and its cluster center \( \text{SumD} \);

Step 5: Use the following way to adjust the cluster center and get the new \( 1^2, \cdots, R^2 \). The coordinate position of the cluster center \( j^2 \) (\( j = 1, \cdots, R \)) is

\[
x_{j^2} = \frac{1}{n_j} \sum_{i \in C_j} x_i, y_{j^2} = \frac{1}{n_j} \sum_{i \in C_j} y_i,
\]

Where \( n_j \) is the number of grid entity in \( C_i \);

Step 6: Repeat Step 3-5 until reach the maximum iteration number \( N \). Output the clustering results corresponding to the minimum value of \( \text{sumD} \);

Step 7: Optimize the result of step 6 by simulated annealing algorithm. The cool way
is \( T(t+1) = k \times T(t) \). In the formula \( k \) is a positive constant slightly less than 1.00 and \( t \) is the times of cooling.

In step 1, the situation that the grid demand is greater than the trusted capacity of grid is considered. In step 2 to 6, cluster the grid s need to, and find the optimal clustering solution. In step 7, the route optimization is done for solving TSP problem.

The clustering process is:

1. Random determine \( R \) (obtained from formula (1)) couples of cluster center;
2. Calculate the trusted value between every grid entity and every cluster center \( d_g(i = 1, \cdots, n; j = 1, \cdots, R) \). Sort \( d_g(j = 1, \cdots, R) \) from small to large and find the smallest trusted value from every grid entity to the cluster center.
3. If the smallest trusted value \( d_g \) is found, then the corresponding grid \( k \) is added to cluster \( p \), and add the grid corresponding to the second smallest value to the corresponding cluster, compute the residual demand \( SuQ \) (that is, the capacity of grid less the amount of grid mounted) of the cluster and turn down. When the residual demand of cluster is less than the demand the grid s want to add, the split entities are selected to split in cluster. The principle of split entity selection will be discussed later.
4. When the total demand of the cluster that the grid s want to join has reached the maximum trusted capacity of grid entity, the second nearest cluster will be considered. Turn down until all grid s are added into a cluster.

In order to ensure the load factor and the least requirement of grid entity, the grid’s need is allowed to split, so the principle of grid choice splitting should be considered. If grid 1 is added into a cluster \( p \) which is not fully loaded, which makes the total demand of the cluster exceeds the maximum trusted capacity of grid entity, the demand needs to be split to meet. If the second nearest cluster center is far away from the grid, the traffic trusted values increase greatly. The unmet demand will be allowed to transmit to a entity whose demand is greater than the unmet demand of grid 1 in cluster \( p \) and which is relatively close to the other cluster whose residual demand should be greater than the unmet demand of grid 1, to make the demand of this entity split meet. The demand of grid 1 is totally met by cluster \( p \). If the residual demand of all clusters is lower than the unmet demand of grid 1, then choose the one with the maximum residual demand to join to avoid being split too many times.

V. COMPARISON OF THE TEST RESULTS OF TLS AND SSL AUTHENTICATION PROTOCOL AND THE NEW GRID AUTHENTICATION METHOD

We set the clouds as a pool with hundreds of computers and there are many grid entities that cannot be trusted or should be limited for intact, then we set some entity to send the request to other grids to compute or calculate some information together, so every grid in the clouds will go into the TLS/SSL model and our new model using distributed parallel authentication model based on trusted computing, then we remind the accuracy and lead time of all the model.

From the table1 and table2 we can see that the accuracy rate of SSL&TLS authentication is lower than distributed parallel authentication model, the lead time of SSL&TLS authentication is longer than distributed parallel authentication model. In table3 we will show the detail comprehensive improvement for different clouds and different internet environment.

<table>
<thead>
<tr>
<th>Experiment index</th>
<th>Accuracy rate(km)</th>
<th>TLS TIME(ms)</th>
<th>SSL TIME(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>63.1</td>
<td>163</td>
<td>537</td>
</tr>
<tr>
<td>2</td>
<td>69.2</td>
<td>175</td>
<td>805</td>
</tr>
<tr>
<td>3</td>
<td>67.9</td>
<td>170</td>
<td>966</td>
</tr>
<tr>
<td>4</td>
<td>66.4</td>
<td>166</td>
<td>881</td>
</tr>
<tr>
<td>5</td>
<td>65.0</td>
<td>169</td>
<td>946</td>
</tr>
<tr>
<td>Aver.</td>
<td>66.32</td>
<td>168.6</td>
<td>827</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Experiment index</th>
<th>Accuracy rate</th>
<th>Computation time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>94.49</td>
<td>5.515</td>
</tr>
<tr>
<td>2</td>
<td>92.12</td>
<td>5.170</td>
</tr>
<tr>
<td>3</td>
<td>94.57</td>
<td>4.911</td>
</tr>
<tr>
<td>4</td>
<td>99.45</td>
<td>5.069</td>
</tr>
<tr>
<td>5</td>
<td>94.74</td>
<td>5.010</td>
</tr>
<tr>
<td>6</td>
<td>91.12</td>
<td>5.053</td>
</tr>
<tr>
<td>7</td>
<td>94.49</td>
<td>5.586</td>
</tr>
<tr>
<td>8</td>
<td>92.12</td>
<td>5.174</td>
</tr>
<tr>
<td>9</td>
<td>94.57</td>
<td>4.938</td>
</tr>
<tr>
<td>10</td>
<td>99.45</td>
<td>5.068</td>
</tr>
<tr>
<td>Aver.</td>
<td>92.712</td>
<td>5.1494</td>
</tr>
</tbody>
</table>

From the table3 we can see the comprehensive evaluation of distributed parallel authentication model is much better than SSL&TLS that the distributed parallel authentication model use less computing operation and computing times but with 30 higher correct accuracy percent and 35.7 equal total percent.
VI. CONCLUSION

From the above analysis, take trusted computing as the basis, in a cloud computing, grid distributed parallel authentication method which is realized by grid authentication and grid behavior simultaneous authentication, established on the upper layer of SSL and TLS protocols, by adaptive stream cipher heuristic code generator and heuristic behavior trust query function, plays well in authentication. However, on the trust issue of grid behavior, further standardization is needed on entities quantitative trust level within a domain, while the core of the heuristic algorithm needs to quantify the grid entities with the shape, weight, size and other physical indicators as a physical entity, this quantitative method still needs to be further improved, so as to promote adaptive stream cipher authentication framework and improve the upper trusted computing platform.

REFERENCES


Keshou Wu (1975.3-), Xiamen city, Fujian Province, China, PhD of Huazhong university of science and technology, majored in software engineering. Research field: System Engineering, Information System, Data Mining, GIS.

Lizhao Liu (1983.3-), Xiamen city, Fujian province, China. PhD candidate of Xiamen university, majored in automation, system engineering, Information Science and Technology Department. Research field: chaotic modeling and control of unmanned airplane vehicle and information system, feature attraction and detection, scale space and multiscale technology. He has done the China national 985 engineering process of unmanned airplane vehicle for the UAV-UAIS chaotic phenomenon analysis, UAV-UAIS chaotic modeling and control. He made the paper such as The Chaotic Characters and New Control Strategy of Unmanned Airplane Information System 2008 ISCID and Error! Reference source not found. The Chaotic Disturbance of UAV System's Communication And Coping Strategy 2008 ICCCAS. He also has done the work of grid behavior trust model and has the paper such as The Quantitative Assignment of The Grid Behavior Trust Model Based on Trusted Computing 2010 Wuhan university journal.Now he is doing the work of scale space and multiscale technology for the image analysis especially for the feature description definition detection and matching.