Journal of Engineering Science and Technology
Vol. 13, No. 12 (2018) 4271 - 4289

LTCHA: LIGHT-WEIGHT TWO-WAY CRYPTOGRAPHIC HASH ALGORITHM FOR CLOUD

V. VASANTHI[1,*], M. CHIDAMBARAM[2]

[1]Research and Development Centre, Bharathiar University, Coimbatore, Tamilnadu, India
[2]Department of Computer Science, Rajah Serfoji Govt. Arts College, Thanjavur, Tamilnadu, India
*Corresponding Author: vasanthi_78@rediffmail.com

Abstract

The cryptographic hash algorithms play a significant role in preserving the data security in the wireless networks. The most frequently used hash algorithms such as Message Digest (MD5) and Secure Hash Algorithm (SHA1) need high computational overhead. The energy-starved network does not afford such high computational overhead. The major constraints in the network are communication, computation and storage overheads. To overcome these issues, this paper develops a Light-Weight Two-Way Cryptographic Hash Algorithm (LTCHA) for generating a hash-digest of minimum length for the energy-starved network. This paper proposes an LTCHA for two-way authentication between the user and Cloud Service Provider (CSP). The Third Party Auditor (TPA) approves the CSP if the identity (ID) and hashcode of the user matches with the generated ID and hashcode. After receiving the approval, the user can upload the file to the service provider. File splitting is performed. After splitting the file, it is allocated to the suitable Virtual Machine (VM) in the cloud by using the ACO-based task allocation algorithm. The proposed LTCHA algorithm requires lower energy consumption, overall execution time, key size, time for encryption and file uploading than the existing schemes.

Keywords: Ant Colony Optimization (ACO), Cloud, Hash algorithm, Light-weight two-way cryptographic hash algorithm, Third party auditor (TPA)

1. Introduction

Hash algorithms play a vital role in the security applications such as node authentication [1, 2], message authentication [3], password protection [4], digital signature [5], etc., in the wireless network. A data string of random length is applied as an input to the hash function and a fixed length string is obtained as output. The hash functions are divided as keyed and unkeyed hash functions whose specification provides two different inputs such as message and a secret key, and a single input parameter such as message respectively. MD5 and SHA1 are the frequently used hash functions. MD5 obtains an input data of random length and generates a message digest of 128 bit long [6]. But, it is not suitable for the applications depending on the Secure Socket Layer (SSL) certificates or digital signatures. Stevens et al. [7] generated X.509 certificates with the similar signatures and dissimilar public keys, based on the MD5 algorithm. The SHA-1 hash function is found to be highly vulnerable [8]. The hash functions including SHA-2 and SHA-3 algorithms are utilized in the government organizations [9, 10].

Akyildiz et al. [11] commented that, in the energy-constrained network with short lifetime, the nodes are arranged in the outdoor environment without requiring human monitoring. Hence, data authentication and reliability are the main concerns to deal with these networks. As energy-constrained network suffers from the constraints such as short battery lifetime, low processing capability and low memory capacity, it cannot be used with the conventional cryptographic algorithms. Hence, it is necessary to develop a lightweight security solution for the Wireless Sensor Network (WSN). Previously, various hash-based security solutions [12, 13] are developed to protect the WSNs from attacks.

Dahmen and Krauß [12] introduced a hash-based signature scheme for authenticating the messages during the unicast and broadcast communication. This scheme yields faster signature generation and verification rate than the existing signature schemes. In the security analysis, it is shown that the signature scheme is preimage resistant. This scheme does not claim the collision resistance. Qin and Chen [13] designed an enhanced hash chain based authentication scheme against the node capture attack in the WSN. The hash function is applied on the preliminary preloaded keys before the deployment of nodes. The leakage of the data due to the attack in the network is reduced owing to the one-way property of the hash function. Barreto et al. [14] proposed a cryptographic hash function that uses large S-boxes on a wide variety of platforms. This function possessed better design than the SHA-3 hash functions. The hash function generates a 512-bit hash digest. But, this hash function is not fast when compared to the existing schemes.

This paper presents an LTCHA that produces a hash digest of fixed and minimum length for two-way authentication between the user and service provider. The proposed LTCHA algorithm satisfies all properties of the two-way unkeyed hash functions. The proposed LTCHA algorithm outperforms the existing authentication schemes in terms of encryption time, energy consumption, file uploading time, execution time and key size.

The following sections in the paper are schematized as: Section 2 describes the existing hash-based authentication schemes used in the wireless network. Section 3 explains the LTCHA algorithm and ACO-based task scheduling algorithm. Section 4 illustrates the performance analysis of the LTCHA algorithm. The conclusion of the proposed work is stated in Section 5.

2. Related Works

Zhuang et al. [15] designed a password authentication scheme without using the smart card. The proposed scheme is based on the geometric hash function. The experimental analysis demonstrated the efficiency of the password authentication scheme against numerous attacks. Das and Goswami[16] presented a secure biometric-based authentication scheme to improve the features of the idle user authentication scheme. The proposed scheme is found to be highly secure against the active and passive attacks. Li et al. [17] suggested a novel authentication scheme depending on the dynamic identity (ID) for the multi-server environments using the smart card. The proposed scheme enabled dynamic change in the user ID for protecting the user from being tracked. He and Wang [18] introduced an authentication scheme based on the biometric technique using the Elliptic Curve Cryptography (ECC) scheme. The proposed authentication scheme achieved a significant reduction in the communication and computational cost. Chen et al. [19] proposed a scheme for password authentication and key agreement for authorizing the users based on the smart card. Better user authentication and high resilience to the stolen smart card attack are achieved. Das [20] developed a new authentication scheme based on the biometric technique for the WSN. The security analysis verified the efficiency of the proposed authentication scheme when compared to the existing security schemes. The drawback of this authentication scheme is its liability to the stolen smart card and forgery attacks.

Li et al. [21] presented a three-factor scheme for authenticating the remote user using the biometric technique to provide more improved security features. This authentication scheme achieved mutual user authentication. Farash and Attari [22] developed a new authentication scheme based on the password for the Session Initiation Protocol (SIP) by using the smart card. An improved authentication scheme is proposed using the elliptic curves. He et al. [23] introduced an identity-based authentication scheme with privacy-preserving property, that does not use bilinear pairing scheme. This scheme ensured simultaneous support to the mutual authentication and privacy protection while requiring low computation and communication cost. Li et al. [24] proposed a new authentication scheme for ensuring secure communication between the customer and neighbor gateway using the Merkle hash tree procedure in the smart grid environment. The proposed scheme achieved a significant reduction in the computation and communication overhead on the smart meters and high resilience to the replay attack. The experimental results demonstrated the efficiency of the proposed scheme with minimum computational complexity. The drawback of this proposed scheme is its great liability to the Denial of Service (DoS) attack.

Al Haddad et al. [25] introduced a Collaborative Network Intrusion Detection System for monitoring the network traffic and detecting the attacks in the cloud computing environment. The proposed system achieved higher attack detection accuracy. Iyengar et al. [26] proposed a security mechanism based on the Fuzzy logic for detecting the malicious packets and mitigating the Distributed DoS (DDoS) attack using appropriate countermeasures. The fuzzy rules are defined based on the traffic pattern of the cloud computing environment. The security mechanism could not protect the data center from all types of DDoS attacks. There is a need to frequently update or modify the security techniques.

A novel authentication scheme for the cloud computing environment is proposed for addressing the forward and backward security problems and provide the anonymity for the user [27]. The user authentication and key exchange are protected using the password, the smart card, and the public key technique. Better security is achieved with the smart card based authentication. But, the computation cost is high. Butun et al. [28] presented a light-weight multi-level authentication service for addressing the scalability problems and time-based constraints in the cloud environment. A large number of devices can be scaled well due to the hierarchical authentication structure. A novel identity (ID) based user authentication scheme for the cloud computing environment is introduced [29]. The scheme is designed using the one-way hash functions and Exclusive OR operations. The proposed scheme required minimum computational and communication cost. An authentication scheme based on the biometric techniques is devised for the multi cloud-server environment. This scheme facilitated the data access from any server upon completion of once-off registration with a valid authority. The anonymity of the user is ensured without requiring expensive operations.

A novel authentication scheme is proposed in the Telecare Medical Information system (TMIS) to offer anonymity, discontinuity and message authentication [30]. This enabled the patients to directly consult with the doctors in a remote location. A set of enhanced and reliable user authentication protocols is introduced over a dynamic cloud environment [31]. The smartphone is transformed for the remote authentication of the TMIS. A strong and privacy-preserving multiple factor authentication scheme is proposed with efficient key distribution mechanism by integrating the fingerprint with the user Identity (ID), password and One-Time Password (OTP) [32]. The main drawback is the user and service provider should trust the third party trustee. A distributed access control scheme is introduced for enabling secure data storage in the cloud environment only by the authenticated users [33]. The authenticity is verified before storing the data in the cloud, without knowing about the user ID. Mahmood et al. [34] developed Two Level Session Key (TKS) based authentication mechanism for Internet of Things (IOT) environment. The proposed scheme provided lightweight and secure communication between the end-to-end users without requiring any third-party authentication sources. From the experimental results, it is concluded that the proposed scheme incurs low communication cost for session key establishment and node association.

3. Proposed Work

In our proposed work, OneDrive is used for storing the files and the personal data of the user in the cloud environment. OneDrive is a file-hosting service operated by the Microsoft Corporation. If the cloud user needs to upload data to the cloud storage, the user has to wait until the CSP approves the data uploading request from the authorized user and sends permission for starting data uploading to the cloud server. The user has to rely on the TPA for auditing the data from the user if the user needs to check the accuracy of data in the cloud environment. The TPA performs auditing to verify the accuracy and integrity of the data, without causing any changes to the original data. In this work, the light-weight hash algorithm is used for the two-way authentication between the user and CSP. The main TPA and a secondary TPA are used for authenticating the user's request.

The secondary TPA responds to the user requests in the queue, while the main TPA responds to the user requests. The hashing scheme generates a hash digest of

small length from an input message of random length. The administrator verifies the registration of the user and approves the registration by sending the user ID, password and Time-based OTP (TOTP) to the user. After receiving the approval of the administrator, the user can login using the ID and TOTP. The hash code is generated for the user by using the LTCHA algorithm. Then, the user submits the file uploading request to the main TPA. The main TPA generates the ID and hashcode and checks whether the ID and hashcode of the user match with the generated ID and hashcode. The main TPA responds to the user request if the ID and hashcode of the user match with the generated ID and hashcode.

The administrator responds the cloud service provider with the user ID, password, and OTP based on the request received from the provider. The service provider can login using the ID and hashcode. The TPA checks whether the ID and hashcode of the service provider match with the generated ID and hashcode. The TPA approves the CSP, if the ID and hashcode match with the generated ID and hashcode. After receiving the approval, the user can upload the file to the service provider. File splitting is performed. In the cloud computing system, scheduling plays a significant role for selecting the optimal resource for executing the tasks. The main aim of task scheduling is to complete the execution of tasks using minimum amount of resources. The scheduling facilitates the CSPs to provide better Quality of Service (QoS). The ACO algorithm is used to find the optimal resource allocation for tasks in the dynamic cloud system to reduce the overall makespan of the system. The allocation of the file to the suitable Virtual Machine (VM) in the cloud is performed using the ACO-based task allocation algorithm. Figure 1 shows the system architecture of our proposed work.

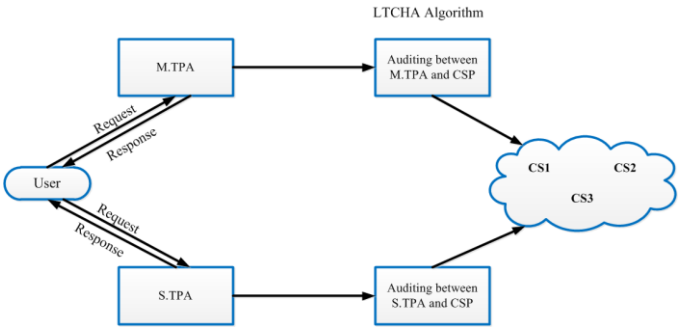

Fig. 1. System architecture of the proposed work.

3.1. LTCHA

The input message includes characters that belong to the set of 96 printable American Standard Code for Information Interchange (ASCII) characters. The input message is pre-processed and converted into the binary representation of the ASCII codes. The definite padding is applied in the Least Significant Bit (LSB) of the input message to make it divisible by 512. An additional 512 zeros are added even if the length of the message is already a multiple of 512, to improve the strength of the algorithm. The pre-processed message is divided three times in a nested manner where the block size of 512-bit is generated in the first level split. The 64-bit and 8-bit block sizes are obtained in the second and third level splits respectively. Three steps of transformations are used on the input message. In the first step of transformation, the 512-bit message block is

divided into 8-bit and 64-bit message blocks. Every 64-bit message block is separated into eight numbers of 8-bit message blocks. Substitution of each 8-bit inner most split block is performed initially by using a prime number selected from a primary substitution table S-Table1 that comprises 97 prime numbers. A number is calculated for each 64-bit message block by using the eight substituted values. The output of the first-level transformation is generated.

After the first transformation is over, the second transformation is performed using a secondary substitution table S-Table2 that consists of 67 prime numbers. The prime numbers are selected in a random way to guarantee uniformity in the transformation process and reduce the storage overhead. The secondary substitution table values are calculated using the formula $\log(\sin(index + 128))$, where $index = 0, \ldots, 66$. The third step transformation is performed using the result of first and second step transformation.

The decimal result of the third step transformation is converted into the equal 3-digit hexadecimal number. The three-level swapping operations are utilized to the 24-digit hexadecimal number of 512-bit message block. The final 64-bit hash digest for every 512-bit message block is received after finishing all the swap operations. The final hash digest for the whole message is obtained by adding all the hash digests for each message block [35]. Figure 2 shows the block diagram of LTCHA algorithm.

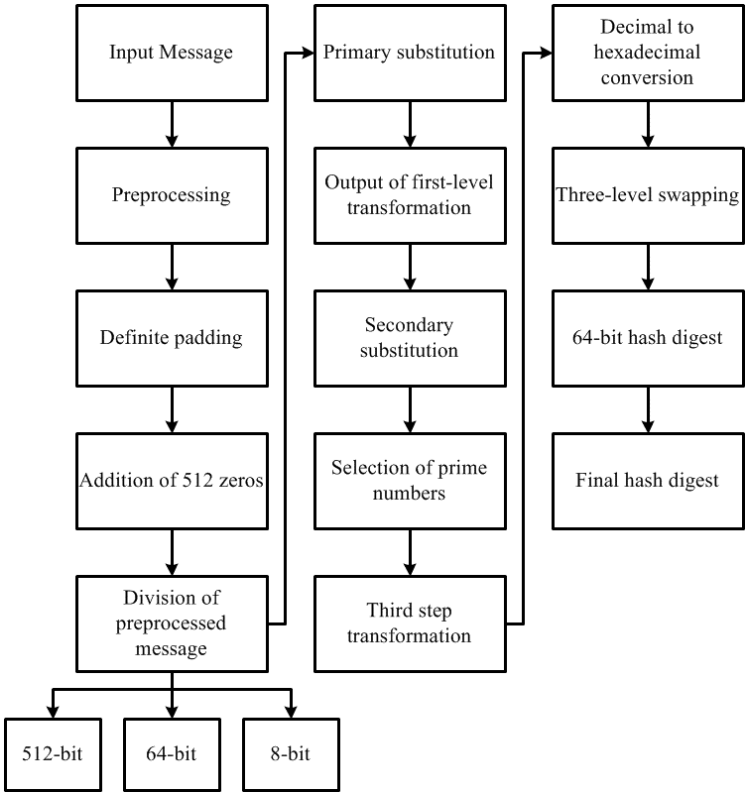

Fig. 2. Block diagram of LTCHA algorithm.

LTCHA

Input: Message (X) of variable length
Output: 64-bit hash-digest (H)
Initialization
1: H to 0//final hash digest
2: First step_conversion of X to 1
3: Second step_$conversion_{part3}$ of X to 7
4: Generate hexadecimal digit $hexdigit_{i(j-1)1} hexdigit_{i(j-1)0}$ to 0 when $j = 0$
Begin
Step 1: Transform each character of X into the 8-bit binary //$X = X_0, X_1, \dots, X_7$
Step 2: Apply definite padding
Step 3: Obtain X' //after adding zeros in the LST of the input message to make it divisible by 512
Step 4: Split X' //Splitting 't' number of 512-bit blocks
Step 5: $for(i = 0; i < t; i++)$ //for 't' number of 512-bit block $subblock_i$
Step 6: $for(j = 0; j \leq 7; j++)$ //for 8 number of 64-bit block $subblock_{ij}$
Step 7: $H_i = \emptyset$ //variable storing the digest for i^{th} 512-bit block is set as empty
Step 8: $for(k = 0; k \leq 7; k++)$ //for 8 number of 8-bit block $subblock_{ijk}$
Step 9: obtain $subblock_{ijk}$ //result of Split X'
Step 10: if $subblock_{ijk}$ contains at least one 1
Step 11: $p_k = decimal(subblock_{ijk}) - 31$ //$p_k \in \{2,3, \dots ,97\}$
Step 12: $subblock_{ijk} = S - Table1[p_k]$
Step 13: else //when $subblock_{ijk}$ contains all zeros
Step 14: $subblock_{ijk} = S - Table1$ //$p_k = 1$
Step 15: end if
Step 16: $Firststep\_conversion = S - Table1(p_k) \times Firststep\_conversion$
Step 17: $if(Firststep\_conversion > 2^{16} - 1)$
Step 18: $Firststep\_conversion = Firststep\_conversion\%2^{16}$
Step 19: end if
Step 20: end for //end of 8 numbers of 8-bit blocks
Step 21: compute $Secondstep\_conversion_{part1} = Firststep\_conversion\%67$
Step 22: if$(subblock_{ijk}[1][7] == 0)$ //$subblock_{ijk}[1][7]$ denotes 8th bit of second subblock$subblock_{ij1}$
Step 23: Compute $Secondstep\_conversion_{part2} = Secondstep\_conversion_{part1}$
Step 24: else
Step 25: Compute $Secondstep\_conversion_{part2} = 67 - Secondstep\_conversion_{part1}$
Step 26: end if
Step 27: Compute $Secondstep\_conversion_{part3} = [Secondstep\_conversion_{part3} + (Secondstep\_conversion_{part1} + Firststep\_conversion)\%256]$

Step 28: Generate $afterSecondstep\_conversion = (Firststep\_conversion\%Secondstep\_conversion_{part3}) + Firststep\_conversion + S - Table2[Secondstep\_conversion_{part2}]$
Step 29: Compute $afterThirdstep\_conversion = (afterSecondstep\_conversion\%255) + p_2 + decimal\ equivalent(hexdigit_{i(j-1)1} hexdigit_{i(j-1)0}) + p_0\%127$ //p_0, p_2 are p_k values when k=0, k=2, respectively
Step 30: Convert to hexnumber$afterThirdstep_{conversion} to\ hexnumber_{ij}$ //$hexnumber_{ij} = hexnumber_{ij0} hexnumber_{ij1} hexnumber_{ij2}$
Step 31: Apply intra-hexnumberhexdigit first-level swapping on each hexnumber
Step 32: Compute $H_i = concat(H_i, hexnumber_{ij})$
Step 33: Apply inter-hexnumberhexdigit second-level swapping on 64-bit 'j' hexdigit
Step 34: end for //completion of 8 and 64-bit blocks
Step 35: Compute $H_i = hexnumber_{i0} hexnumber_{i1} \dots \dots hexnumber_{i6} hexnumber_{i7}$
Step 36: Apply inter-hexnumber third-level swapping on the hexnumberH_i
Step 37: Calculate final hash digest $H = H + H_i$ //ignore the carry bits
Step 38: end for //completion of 512-bit blocks

3.2. ACO-based Task Scheduling

After splitting the file, it is allocated to the suitable VM in the cloud by using the ACO-based task allocation algorithm. The main advantage of the ACO algorithm over other meta-heuristic algorithms is the dynamic change in the problem instance. The decisions made by all ants are purposeful and output of all ants is utilized to construct the new optimal solution, during each iteration. The best solution is obtained by determining the changes in the pheromone concentration [36]. As ACO can be used for solving NP-hard problems such as traveling salesman problem [37], graph coloring problem [38], vehicle routing and scheduling problems [39], it is highly appropriate for dynamic task scheduling in cloud [40].

The ACO algorithm is developed according to the foraging activities of the ant colonies. The group of ants uses a special chemical known as pheromone to communicate with other ants, while searching for the food. Initially, the ants randomly start searching for the food. Once the ants find a path for the food, they leave pheromone along that path, so that the ants can easily follows the path by sensing the pheromone. Most of the ants are attracted to the shortest path [41]. The ACO-based task scheduling algorithm is presented to reduce the overall makespan of the tasks [42].

3.2.1. Initialization of Pheromone

The virtual pheromone path $\tau_{ij(t)}$ on the edge connects the task *'i'* to the VM_j. The preliminary amount of the pheromone on the edges is estimated to be a minor positive constant τ_0.

3.2.2. Rule for Choosing VM for the next task

During each iteration, each ant *'k'* $where\ k = 1, 2, \dots, m$ (m represents the number of the ants) creates a path that executes *'n'* number of steps. A probabilistic

transition rule is applied to the ant. The *k*-ant chooses the VM (VM_j) for the subsequent task *'i'* with the probability calculated by using the following equation

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{s \in allowed_k} [\tau_{is}(t)]^{\alpha} * [\eta_{is}]^{\beta}} \; if \; j \in allowed_k \\ 0 \qquad\qquad\qquad\qquad \text{Otherwise} \end{cases} \tag{1}$$

where $\tau_{ij(t)}$ shows the pheromone concentration at the time 't' on the pheromone path between the task *'i'* and VM_j. This allows $k = \{0,1,\ldots,n-1\} - tabu_k$ represents the allowed VMs for the ant *'k'* in the next step. $tabu_k$ denotes the traversed VM by the ant *'k'* and $\eta_{ij} = 1/d_{ij}$ signifies the visibility for the *'t'* moment, computed using the heuristic algorithm. d_{ij} denotes the estimated execution time and transfer time of the task on the VM. It is calculated as

$$d_{ij} = \frac{TL\_Task_i}{Pe\_num_j * Pe\_mips_j} + \frac{InputFileSize}{VM\_bw_j} \tag{2}$$

where TL_Task_i represents the total length of the task submitted to the VM, Pe_num_j denotes the number of processors in the VM and Pe_mips_j indicates the Million Instructions Per Second (MIPS) of each processor. $InputFileSize$ denotes the size of the task before execution and VM_bw_j represents the bandwidth capability of the VM_j. α and β controls the relative weight of the path and visibility information.

3.2.3. Pheromone Update

After completing the path, each ant *'k'* places the amount of pheromone $\Delta\tau_{ij}^{k}(t)$ computed on each edge (i, j)

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L^k(t)} \; if \; (i,j) \in T^k(t) \\ 0 \quad if \; (i,j) \notin T^k(t) \end{cases} \tag{3}$$

where $T^k(t)$ represents the path taken by the ant *'k'* at the time *'t'* and *Q* denotes an adaptive parameter. $L^k(t)$ denotes the length of the pheromone path computed as

$$L^k(t) = \arg\max_{j \in J} \{sum_{i \in IJ}(d_{ij})\} \tag{4}$$

where *ij* denotes the set of the tasks that are allocated to the VM. After updating the pheromone, it is applied to all edges as

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{5}$$

where ρ is the path degeneration $0 < \rho < 1$. $\Delta\tau_{ij}(t)$ is computed as

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta_{\tau_{ij}}^{k}(t) \tag{6}$$

When all the ants complete the entire path, an ant highlights the pheromone on the edges that belong to the best path created from the beginning of the path (T^+) by the amount Q/L^+. L^+ represents the length of the best path. This is termed as global pheromone update.

$$\tau_{ij}(t) = \tau_{ij}(t) + Q/L^+ \quad if \; (i,j) \in T^+ \tag{7}$$

3.2.4. Task scheduling

The artificial ant stochastically assigns each task to one processor until each task is assigned to specific processor. An artificial pheromone value $\tau_{i,j}$ indicates the favourability of assigning the task T_i to the processor P_j. Initially the artificial pheromone value is same for all tasks and processor units. After each iteration, the pheromone value of each edge is minimized to emulate the real-life behaviour of evaporation of the pheromone count with respect to time. ρspecifies the percentage of the artificial pheromone value after evaporation. 1-τ represents the evaporation rate. Each ant behaves as follows: From a node *'i'* in the task set, an ant chooses a node *'j'* in the processors with a probability given by:

$$\rho(i,j) = \frac{\tau_{i,j}}{\sum_{j=1}^{m} \tau_{i,j}} \tag{8}$$

After all the tasks are scheduled by an ant, the feasibility of the schedule is verified using the utilization rate of the individual processors. If the utilization rate of any processor exceeds 1.0, that schedule is infeasible. This procedure is repeated for all ants. The quality *'q'* of a feasible schedule '*S*' generated by an ant is computed by considering the total utilization rate of all processors. This quality is used in the pheromone update of the next iteration [43].

ACO algorithm

Step 1: Initialization
  Set Current_iteration_*t*=1
  Set Current_optimal_solution=null
  Set Initial value $\tau_{ij}(t) = c$ for each path between the tasks and VMs
Step 2: Place *'m'* ants on the starting VMs in a random manner
Step 3: For *k*=1 *to m* do
  Arrange the starting VM of the k^{th} ant in $tabu_k$
  Do ants_trip while all the ants do not end their trips
  Every ant chooses the VM for the next task according to eqn (1)
  Insert the selected VM to $tabu_k$
  End Do
Step 4: For *k*=1 *to m* do
  Calculate the length L_k of the tour defined by the k^{th} ant according to eqn (4)
  Update the current_optimal_solution with the best solution
Step 5: Apply the local pheromone for every edge (i,j) based on eqn (5)
Step 6: Update global pheromone according to eqn (7)
Step 7: Increment Current_iteration_*t* by one
Step 8: If $(Current\_iteration\_t < t_{max})$
  Empty all tabu lists
  Goto Step 2
  Else
  Print current_optimal_solution
  End If
Step 9: Return

ACO algorithm based task scheduling

Input: Incoming Tasks and list of VMs

Output: Scheduling tasks
Step 1: Set Task_List=null and temp_List_of_Cloudlet=null
Step 2: Place any incoming tasks in the Task_List in order of the arrival time of the tasks
Step 3: Do ACO_P while Task_List not empty or there are more incoming tasks
    Set *n*=size of VMs list
    If (size of Task_List greater than *'n'*)
        Transfer the first arrived *'n'* tasks from Task_List and place them on temp_List_of_Tasks
    Else
        Transfer all tasks from Task_List and place them on temp_List_of_Tasks
    End If
        Execute ACO procedure with input temp_List_of_Tasks and *n*
    End Do
Step 4: Print "task scheduling completed"
Step 5: Stop

4. Performance Analysis

The proposed LTCHA algorithm is analyzed using the Intel (R) Pentium (R) CPU G2010 @ 2.80GHz 2.80 GHz processor with 4GB memory. The files and the personal data of the user are stored in OneDrive. The input for the LTCHA algorithm is the message of variable length and a 64-bit hash digest is obtained as the output. The input message is divided into 8-bit, 64-bit and 512-bit. A 64-bit key is used in this proposed work. The proposed scheme consumes energy of about 39 microjoules. Table 1 shows the experimental setup of the proposed work.

Table 1 Experimental setup.

File hosting service	OneDrive
Input message	Message of variable length
Output message	64-bit hash-digest (H)
Division of Input Message	3 times [512bit, 64 bit, 8 bit]
Key Size	64 bit
Energy Consumption	39 microjoules

The proposed LTCHA algorithm is compared with the Light-weight Hybrid Key Management Scheme (LHKMS), Advanced Encryption Standard (AES), ECC, Two-level Session Key (TSK), SHA-1 and Elliptic Curve Digital Signature Algorithm (ECDSA) algorithms [34]. The proposed LTCHA algorithm is compared with the Light-weight Hybrid Key Management Scheme (LHKMS). Figure 3 shows the encryption time analysis of the proposed LTCHA and LHKMS. There is a linear increase in the encryption time with respect to the increase in the number of characters. The proposed LTCHA requires minimum encryption time than the LHKMS scheme. The efficiency of the proposed scheme is analyzed by comparing the key size and energy consumption of the LHKMS and existing AES, ECC, Two-level Session Key (TSK), Elliptic Curve Digital Signature Algorithm (ECDSA) and SHA-1 schemes. Figure 4 shows the impact of the key sizes and Fig. 5 shows the energy consumption for the proposed and existing schemes. The key size of the proposed LTCHA algorithm is 64 bits and energy consumption is 39 microJoules.

The key size of the LHKMS scheme is 55 bits and energy consumption is 41 microJoules. The TSK scheme uses a 64 bit key and consumes 46.08 microJoules. The AES scheme uses a 128 bit key and consumes 92.16 microJoules. The ECC scheme uses a 108 bit key and consumes 77.76 microJoules and ECDSA scheme uses a 160 bit key and consumes 115.2 microJoules. The SHA-1 algorithm uses a 128 bit key and consumes 92.16 microJoules. From the graph, it is observed that the key size and energy consumption of the proposed LTCHA algorithm is lower than the LHKMS scheme, TSK, AES, ECC, ECDSA and SHA-1 algorithms. Figure 6 illustrates the time taken by the proposed LTCHA algorithm and existing Rivest-Shamir-Adleman (RSA) algorithm, advanced secret key sharing management scheme [44] and LHKMS scheme for file downloading for different file sizes.

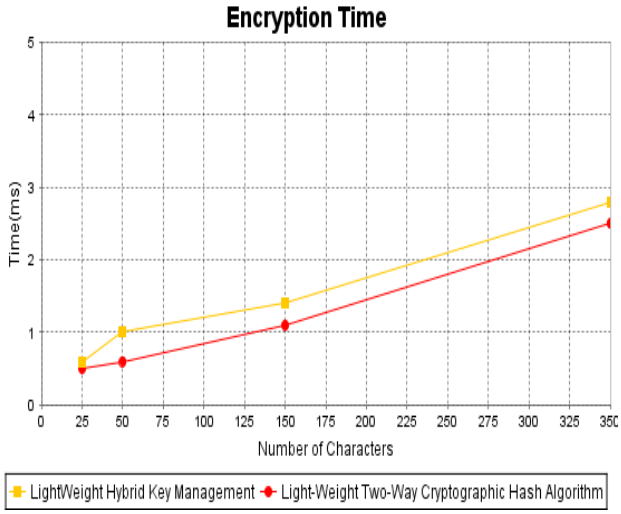

Fig. 3. Encryption time analysis of the proposed LTCHA and LHKMS.

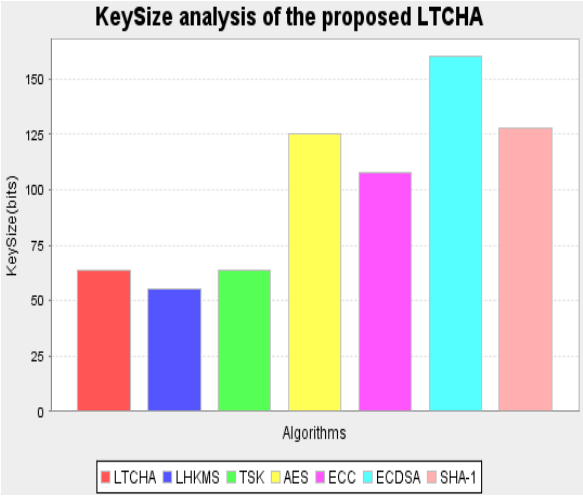

Fig. 4. Key size analysis of the proposed LTCHA algorithm and LHKMS, TSK, AES, ECC, ECDSA and SHA-1 algorithms.

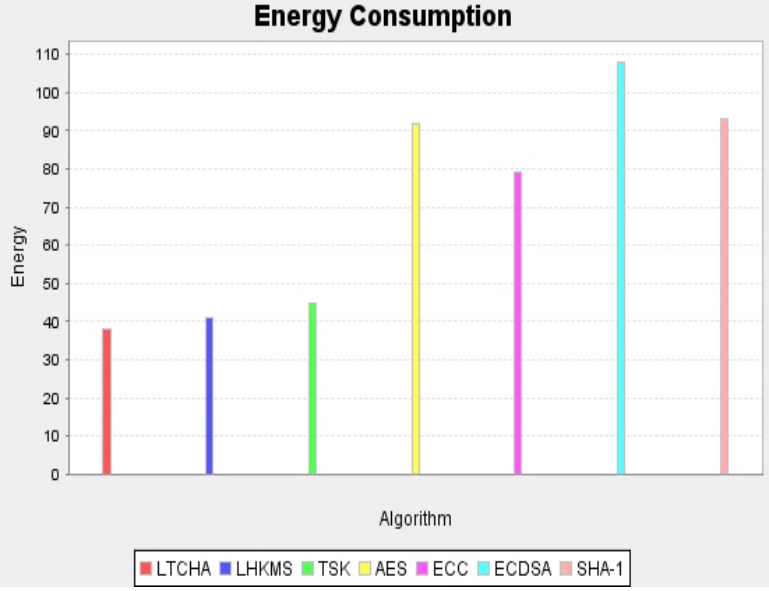

Fig. 5. Energy consumption of the proposed LTCHA and LHKMS, TSK, AES, ECC, ECDSA and SHA-1 algorithms.

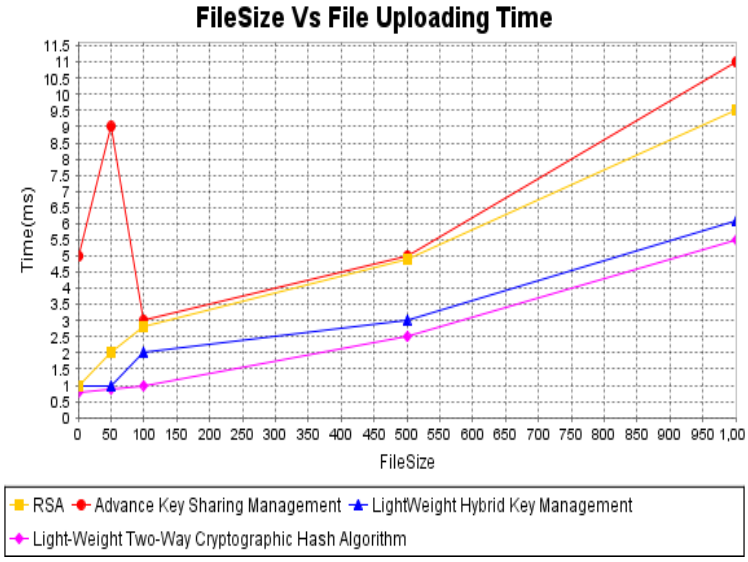

Fig. 6. File uploading time analysis of the proposed LTCHA algorithm and existing RSA, advanced secret key sharing management scheme and LHKMS scheme.

Our proposed LTCHA algorithm is compared with Price-based Maxmin [45], Fair Allocation [46, 47] and Trust-based Maxmin [48]. The Price-based Maxmin derives the distribution of optimal detection load. The fair allocation model distributes the detection load equally among all the VMs in a fair way. The trust scores of the VMs are considered in the trust-based maxmin model. Figure 7 shows

the execution time analysis of the proposed ACO-based task allocation algorithm and existing fair allocation, price based maxmin and trust-based maxmin schemes.

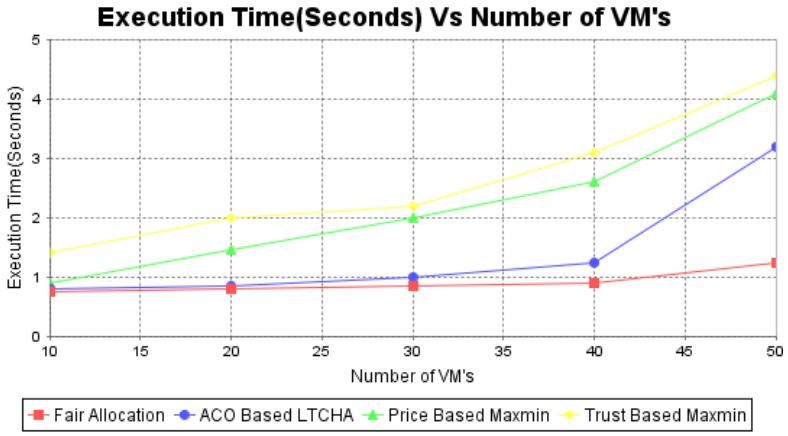

Fig. 7. Execution time analysis of the proposed algorithm and existing fair allocation, price based maxmin and trust-based maxmin schemes.

False Positive Rate (FPR) represents the number of incorrect detected results to the total number of incorrect detection results. False Negative Rate (FNR) is the ratio of positive outcomes that yield negative detection outcomes. Our proposed scheme is evaluated using the fingerprint-based authentication FVC2006 database. Figure 8 shows the FPR analysis of the proposed LTCHA with the existing works and Multi-Factor Authentication (MFA) scheme [32]. Figure 9 depicts the FNR analysis of the proposed LTCHA with the existing fingerprint schemes. From the graphs, it is concluded that the proposed LTCHA algorithm yields minimum FNR than the existing schemes.

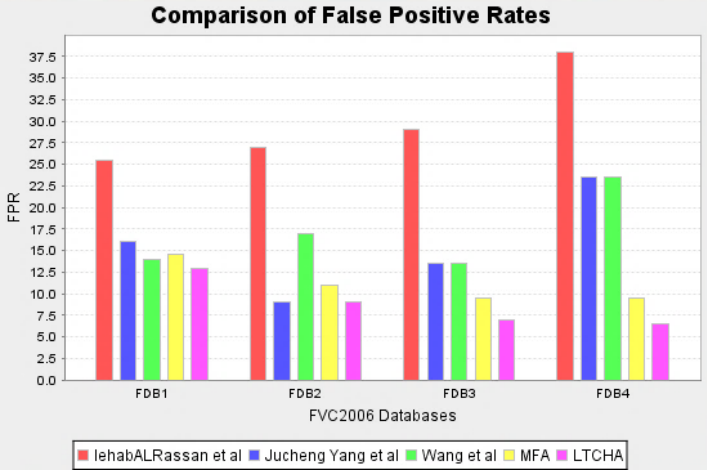

Fig. 8. FPR analysis of LTCHA algorithm and fingerprint based authentication schemes

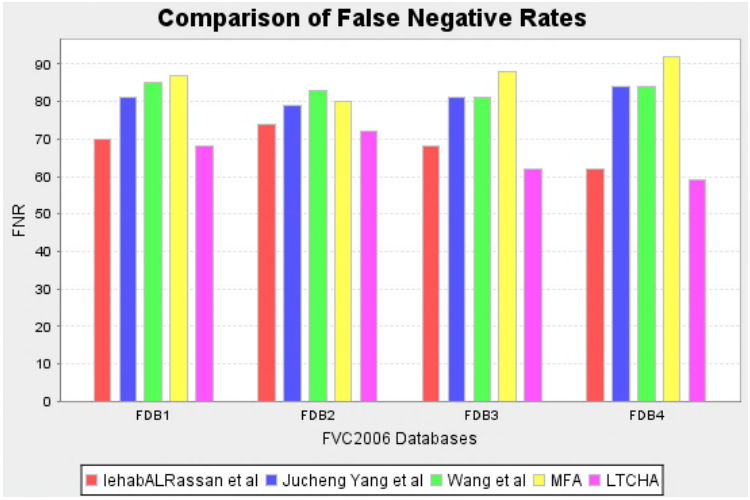

Fig. 9. FNR analysis of LTCHA algorithm and fingerprint based authentication schemes.

5. Conclusions

This paper presents a LTCHA algorithm that generates a hash digest of a small length for improving the security in the wireless network. The proposed algorithm consumes minimum amount of communication, computational and storage overhead due to the light-weight key. The energy efficiency of the proposed scheme is higher than the SHA-1 and MD5 hash functions. The proposed LTCHA algorithm is compared with the LHKMS, AES, ECC, TSK, ECDSA, SHA-1 algorithms, RSA and advanced secret key sharing management scheme. From the experimental analysis, it is concluded that the proposed LTCHA algorithm consumes minimum encryption scheme, energy consumption, key size, file uploading time and execution time than the existing schemes. The key size of the proposed LTCHA algorithm is 64 bits and energy consumption is 39 MicroJoules. Hence, the proposed LTCHA algorithm is found to be highly efficient than the existing algorithms.

Abbreviations

ECDSA	Elliptic Curve Digital Signature Algorithm
ID	Identity
LHKMS	Light-weight Hybrid Key Management Scheme
TPA	Third Party Auditor
TSK	Two-level Session Key
VM	Virtual Machine

References

1. Zhang, Y.; Yang, J.; Li, W.; Wang, L.; and Jin, L.; (2010). An authentication scheme for locating compromised sensor nodes in WSNs. *Journal of Network and Computer Applications,* 33(1), 50-62.
2. Dong, X.; and Li, X.; (2009). An authentication method for self nodes based on watermarking in wireless sensor networks. *Proceedings of the 5th*

International Conference on Wireless Communications, Networking and Mobile Computing. Beijing, China, 1-4.

3. Yu, C.-M.; Tsou, Y.-T.; Lu, C.-S.; and Kuo, S.-Y.; (2011). Constrained function-based message authentication for sensor networks. *IEEE Transactions on Information Forensics and Security*, 6(2), 407-425.
4. Das, M.L.; Saxena, A.; and Gulati, V.P.; (2004). A dynamic ID-based remote user authentication scheme. *IEEE Transactions on Consumer Electronics*, 50(2), 629-631.
5. Jeng, F.-G; Chen, T.-L.; and Chen, T.-S.; (2010). An ECC-based blind signature scheme. *Joural of Networks*, 5(8), 921-928.
6. Wang, X.; and Yu, H.; (2005). How to break MD5 and other hash functions. *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. *Denmark*, 19-35.
7. Stevens, M.; Lenstra, A.; and de Weger, B.; (2007). Chosen-prefix collisions for MD5 and colliding X. 509 certificates for different identities. *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Barcelona, Spain, 1-22.
8. Rijmen, V.; and Oswald, E.; (2005). Update on SHA-1. *Proceedings of the Cryptographers' Track at the RSA Conference. San Francisco, California*, 58-71.
9. McEvoy, R.P.; Crowe, F.M., Murphy, C.C.; and Marnane, W.P.; (2006). Optimisation of the SHA-2 family of hash functions on FPGAs. *IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures.* Karlsruhe, Germany, 6 pages.
10. Jararweh, Y.; Tawalbeh, L.; and Moh'd, A.; (2012). Hardware performance evaluation of SHA-3 candidate algorithms. *Journal of Information Security*, 3(2), 69-76.
11. Akyildiz, I.F.; Su, W.; Sankarasubramaniam, Y.; and Cayirci, E.; (2002). Wireless sensor networks: A survey. *Computer networks*, 38(4), 393-422.
12. Dahmen, E.; and Krauß, C.; (2009). Short hash-based signatures for wireless sensor networks. *Proceedings of the International Conference on Cryptology and Network Security*. Kanazawa, Japan, 463-476.
13. Qin, T.; and Chen, H.; (2012). An enhanced scheme against node capture attack using hash chain for wireless sensor network. *Information Technology Journal*, 11(1), 102-109.
14. Barreto, P.; Nikov, V.; Nikova, S.; Rijmen, V.; and Tischhauser, E.; (2010). Whirlwind: A new cryptographic hash function. *Designs, Codes and Cryptography*, 56(2-3), 141-162.
15. Zhuang, X.; Chang, C.-C.; Wang, Z.-H.; and Zhu, Y.; (2014). A simple password authentication scheme based on Geometric Hashing function. *International Journal of Network Security*, 16(3), 237-243.
16. Das, A.K.; and Goswami, A.; (2014). An enhanced biometric authentication scheme for telecare medicine information systems with nonce using chaotic hash function. *Journal of Medical Systems*, 38(6), Article 27.
17. Li, X.; Ma, J.; Wang, W.; Xiong, Y.; and Zhang, J.; (2013). A novel smart card and dynamic ID based remote user authentication scheme for multi-server environments. *Mathematical and Computer Modelling*, 58(1-2), 85-95.

18. He, D.; and Wang, D.; (2015). Robust biometrics-based authentication scheme for multiserver environment. *IEEE Systems Journal*, 9(3), 816-823.
19. Chen, B.-L.; Kuo, W.-C.; and Wuu, L.-C.; (2014). Robust smart-card-based remote user password authentication scheme. *International Journal of Communication Systems*, 27(2), 377-389.
20. Das, A.K.; (2017). A secure and effective biometric-based user authentication scheme for wireless sensor networks using smart card and fuzzy extractor. *International Journal of Communication Systems*, 30(1), 25 pages.
21. Lin, W.; Liang, C.; Wang, J.Z.; and Buyya, R.; (2014). Bandwidth-aware divisible task scheduling for cloud computing. *Software: Practice and Experience*, 44(2), 163-174.
22. Farash, M.S.; and Attari, M.A.; (2016). An anonymous and untraceable password-based authentication scheme for session initiation protocol using smart cards. *International Journal of Communication Systems*, 29(13), 1956-1967.
23. He, D.; Zeadally, S.; Xu, B.; and Huang, X.; (2015). An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks. *IEEE Transactions on Information Forensics and Security*, 10(12), 2681-2691.
24. Li, H.; Lu, R.; Zhou, L.; Yang, B.; and Shen, X.; (2014). An efficient merkle-tree-based authentication scheme for smart grid. *IEEE Systems Journal*, 8(2), 655-663.
25. Al Haddad, Z.; Hanoune, M.; and Mamouni, A. (2016). A collaborative framework for intrusion detection (C-NIDS) in Cloud computing. *Proceedings of the International Conference on Cloud Computing Technologies and Applications (CloudTech)*. Marrakech, Morocco, 261-265.
26. Iyengar, N.C.S.N.; Banerjee, A.; and Ganapathy, G.; (2014). A fuzzy logic based defense mechanism against distributed denial of service attack in cloud computing environment. *International Journal of Communication Networks and Information Security (IJCNIS)*, 6(3), 233-245.
27. Li, H.; Li, F.; Song, C.; and Yan, Y.; (2015). Towards smart card based mutual authentication schemes in cloud computing. *KSII Transactions on Internet and Information Systems*, 9(7), 2719-2735.
28. Butun, I.; Erol-Kantarci, M.; Kantarci, B.; and Song, H.; (2016). Cloud-centric multi-level authentication as a service for secure public safety device networks. *IEEE Communications Magazine*, 54(4), 47-53.
29. Yang, J.H.; and Lin, P.Y.; (2014). An ID-based user authentication scheme for cloud computing. *Proceedings of the Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. Kitakyushu, Japan, 98-101.
30. Chiou, S.-Y.; Ying, Z.; and Liu, J.; (2016). Improvement of a privacy authentication scheme based on cloud for medical environment. *Journal of Medical Systems*, 40(4), 1-15.
31. Siddiqui, Z.; Abdullah, A.H.; Khan, M.K.; and Alghamdi, A.S.; (2014). Smart environment as a service: Three factor cloud based user authentication for telecare medical information system. *Journal of medical systems*, 38(1), 1-14.

32. Nagaraju, S.; and Parthiban, L.; (2016). SecAuthn: Provably secure multi-factor authentication for the cloud computing systems. *Indian Journal of Science and Technology*, 9(9), 18 pages.
33. Ruj, S.; Stojmenovic, M.; and Nayak, A.; (2014). Decentralized access control with anonymous authentication of data stored in clouds. *IEEE transactions on Parallel and Distributed Systems*, 25(2), 384-394.
34. Mahmood, Z.; Ning, H.; and Ghafoor, A.; (2016). Lightweight two-level session key management for end user authentication in Internet of Things. *Proceedings of the IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data.* Chengdu, China, 323-327.
35. Chowdhury, A.R.; Chatterjee, T.; and DasBit, S.; (2014). LOCHA: a light-weight one-way cryptographic hash algorithm for wireless sensor network. *Procedia Computer Science*, 32, 497-504.
36. Vinothina, V.; and Sridaran, R. (2014). Survey-ACO in task scheduling problem. *International Journal of Advanced Networking Applications*, 132-136.
37. Dorigo. M.; and Gambardella, L.M.; (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1), 53-66.
38. Salari, E.; and Eshghi, K.; (2005). An ACO algorithm for graph coloring problem. *Proceedings of the ICSC Congress on Computational Intelligence Methods and Applications*, Istanbul, Turkey, 5 pages.
39. Zhang, X.; and Tang, L.; (2005). CT-ACO-hybridizing ant colony optimization with cyclic transfer search for the vehicle routing problem. *Proceedings of the ICSC Congress on Computational Intelligence Methods and Applications*. Istanbul, Turkey, 6 pages.
40. Guo, Q. (2017). Task scheduling based on ant colony optimization in cloud environment. *AIP Conference Proceedings*, 1834(1), 11 pages.
41. Dorigo, M.; Birattari, M.; and Stutzle, T.; (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4), 28-39.
42. Tawfeek, M.A.; El-Sisi, A.; Keshk, A.E.; and Torkey, F.A.; (2013). Cloud task scheduling based on ant colony optimization. *Proceedings of the 8th International Conference on Computer Engineering & Systems (ICCES)*. Cairo, Egypt, 64-69.
43. Srikanth, U.G.; Maheswari, V.U.; Shanthi, P.; and Siromoney, A.; (2012). Tasks scheduling using ant colony optimization. *Journal of Computer Science*, 8(8), 1314-1320.
44. Mishra, A.; (2014). *Data security in cloud computing based on advanced secret sharing key management scheme.* Master Thesis. Department of Computer Science and Engineering, National Institute of Technology, Rourkela.
45. Wahab, O.A.; Bentahar, J.; Otrok, H.; and Mourad, A.; (2016). How to distribute the detection load among virtual machines to maximize the detection of distributed attacks in the cloud. *Proceedings of the IEEE International Conference on Services Computing*. San Francisco, California, United States of America, 316-323.

46. Wang, W.; Liang, B.; and Li, B.; (2015). Multi-resource fair allocation in heterogeneous cloud computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 26(10), 2822-2835.
47. Wei, G.; Vasilakos, A.V.; Zheng, Y.; and Xiong, N.; (2010). A game-theoretic method of fair resource allocation for cloud computing services. *The Journal of Supercomputing*, 54(2), 252-269.
48. Wahab, O.A; Bentahar, J.; Otrok, H.; and Mourad, A.; (2018). Optimal load distribution for the detection of VM-based DDoS attacks in the Cloud. *IEEE Transactions on Services Computing*, 1, 1-14.