International Journal of Security and Its Applications
Vol. 9, No. 5 (2015), pp. 85-94
http://dx.doi.org/10.14257/ijsia.2015.9.5.09

A Secure and Fault Tolerant Platform for Mobile Agent Systems

Rajdeep Bhanot[1] and Rahul Hans[2]

[1]Research scholar, DAV University Jalandhar
[2]Assistant Professor in dept. of CSE, DAV University Jalandhar
Email: er.rajbhanot@gmail.com, rahulhans@gmail.com

Abstract

Mobile agents offer a new paradigm in the field of distributed computing. Mobile agent is a program which can migrate from one machine to another machine in order to fulfill the client's needs. Since it moves from machine to machine on the demand of client, there is threat to the data it carries, from the malicious node or hacker who can steal or change the confidential data of the client. In this paper, we have proposed an integrated framework which is fault tolerant as well as secure from such malicious nodes or hackers. We have applied encryption algorithm for data security and fault tolerant mechanism to avoid any kind of fault using clone and check-pointing with location tracking mechanism. We have implemented the proposed approach and evaluated the time taken by the agent on the basis of various parameters with its fault tolerant and security feature.

Keywords: *Mobile agent, Security, Blowfish, Encryption, Fault tolerant, clone, check-pointing*

1. Introduction

An agent is a program that assists people and acts on their behalf. These are actually mobile autonomous processes which act for the client needs from server to server [5]. Agents are basically of two types:

- Stationary agent
- Mobile agent

Stationary agents are the programs which acts on the behalf of user but these agents cannot move from server to server. These agents just sit on one server and communicate with its surroundings by conventional means, such as various forms of remote procedure calling (RPC) and messaging. These stationary agents executes only on the system where it begins its execution.

Mobile agents are also the smart programs which act on the behalf of user but, these are not bound to the system where these start their execution. These have the unique ability to migrate itself from one system to another system in a network. These are the smart programs which are dispatched from home server and can migrate from one host to another host in a network to accomplish the task given to it.

Basically Mobile agent is a movable software program that acts on the behalf of user or client to retrieve and process information. It execute persistently rather than on demand. Only agent decides when to take action and what to do. Mobile agent itself decide when and where to move [3].

ISSN: 1738-9984 IJSIA
Copyright ⓒ 2015 SERSC

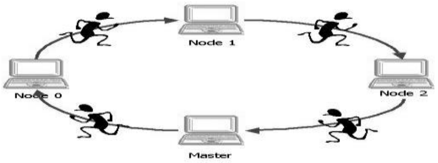

Figure 1. Agent Movement

It communicates in agent communication language. Mobile agent has the following unique and important characteristics like *Object passing*, *Autonomous, Asynchronous, Local interaction, disconnected operation, and Parallel execution.* These programs works on java based Aglets platform.

Aglet is a java based mobile agent platform and library for building mobile agent based applications. Aglet uses J-AAPI and Aglet object model for aglet programming. It uses Aglet transfer protocol (ATP). ATP is an application level protocol for distributed agent based system [13].

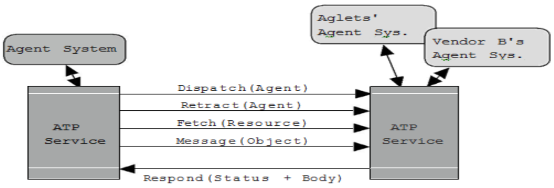

Figure 2. Aglet Transfer Protocol

ATP defines four standard request methods for agent services: dispatch, retract, fetch and message. Now, in the movement of agents there are some threats to the data it carries. So agent should be secure and fault tollerent to avoid any kind of threat, whether it is security threat from some intruder or it's disposal threat from faulty node. Faullty nodes can destroy the agent and the data it carrying. Mobile agents can be secured against these faulty nodes by using fault tollerent machenism that uses the concept of clone and check-pointing. But security is also a big issue in this paradiagm, because in an agent itenrery agent may came across malicious node or host which can damage the confidential and important data. When agent move from one host to another while carrying data, it may face some attack on its data from any intruder or malicious agent. So we also have to make it secure against such type of attacks.

Security in mobile agents is classified as agent security and host security. Agent security is further devided into code security and data security. Code security methos contain the code security machanism like Code obfuscation whereas, data security method contain the security of data that agent carries.

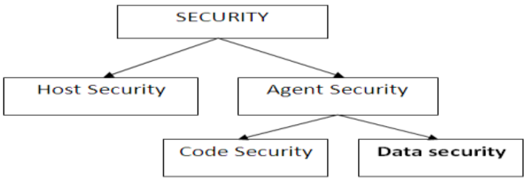

Figure 3. Security Types

In this paper, we are going to take data security in under our consideration. Data security mechanism will provide security to data that agent carries from node to node. We will achieve security by using encryption algorithm. We will send the encrypted data from home node to destination node and there at other end(destination node) , decryption of data will occur.

2. Analysis of Related Work

Mobile agent infrastructure is very useful for us in today's software systems but its two weak sides, security and fault tolleration are the darkest sides where some hacker or malicious node can harm our important data without our permission. Because, as we dispatch the mobile agent from home node to the destination node for some perticular task, it gets out of host control and if there is some malicious or faulty environment it may lead to destroy of agent and its data. There are some threats to mobile agent from faulty and malicious environment:

- Agent disposal
- Agent code modification
- Agent data manipulation
- Analysis of original data collected by an agent

Above threats are from malicious or faulty nodes. So to avoid these common threats some researchers have found there solutions according to their resources, even then there are some threats left to security of agent and fault tolleration. We will discuss here what previous researchers have implemented regarding these problems. First we will discuss about fault tollerance in mobile agent infrastructure. We will discuss about some previously proposed fault tollerant techniques.

F. Alan [8] described that Agent replication is the act of creating one or more duplicates of one or more agents in a multi-agent system. In this author proposed that using redundancy by replication of individual agents within a multi-agent system is one possible approach for improving fault-tolerance. Each of these duplicates is capable of performing the same task as the original agent. The group of duplicate agents is referred to as a replicate group and the individual agents within the replicate group are referred to as replicates. Once a replicate group is created in a mobile agent system, if that replicate group is visible to the rest of the mobile agent system, there are several ways that agents can interact with it by sending messages. The advantage of this method is that an acceptable degree of overhead is imposed by replication on the system and it improved system reliability. It can be both blocking and non-blocking. Communication, read/write consistency and state synchronization issues can be catered as well by the technique using transparent proxies.

A. Budi [11] In this paper, author introduced the CAMA the Context-Aware Mobile Agents framework which supports application-level fault tolerance by providing a set of abstractions and a supporting middleware that allow developers to design elective error detection and recovery mechanisms. CAMA supports system fault tolerance through exception handling and structured agent coordination. There are three basic operations available to the CAMA agents for catching and raising inter-agent exceptions raise, check and wait. These functionalities are complementary and orthogonal to the application level mechanism used for programming internal agent behavior. The advantage of this approach is that the exception handling allows fast and effective application recovery by supporting flexible choice of the handling scope and of the exception propagation policy and also it deals with agent's failures and connection disconnection problems. Its drawback is that it can be blocking in the case when an exception is raised to the agent which has left the scope.

S. Bagchi [9] described that Chameleon provides an adaptive infrastructure that supports different levels of availability requirements simultaneously in a single heterogeneous environment. The advantage of this approach is that provides a flexible architecture through which adaptive fault tolerance may be achieved in an unreliable and heterogeneous network and it deals with both agent and system failure. It has a disadvantage that it suffers from blocking if any of the nodes fails during execution.

K. Damasceno [12] described that MoCA is a middleware system supporting development and execution of the context-aware collaborative applications which work with mobile users. In the author proposes a novel context-aware exception handling mechanism. It has three elements that compose the MoCA application: a server, a proxy, and clients. The first two are executed on the nodes of the wired network, while the clients run on mobile devices. Author has implemented error handling features in several prototype context-aware collaborative applications built with the MoCA. Author also describes the prototype implementation of the model in the MoCA middleware; it consists of an extension of the client and server APIs and new middleware services, such as management of exceptional contexts. The advantage of this approach is that MoCA copes with agent’s failure using both backward and forward recovery and also user transparency is achieved. Coordination among the agents is direct within scopes. Author also claims it to be dependable and efficient technique.

S.J. Choi [10] has discussed region based stage construction protocol which is used for fault tolerant execution of mobile agents in a multi-region mobile agent computing environment. It uses new concepts of quasi-participant and sub stage in order to put together some places located in different regions within a stage in the same region. A mobile agent a_i executes tasks on a sequence of nodes. Each action that a_i execute on a place p_i is called a step each step consists of a set of places called a stage S_i . $p^w{}_i$ at S_i is called a worker, the others are called participants. When a worker fails, one of participants is elected as a new worker and takes over the action of the previous worker. To provide the exactly once property of a mobile agent execution, voting and agreement protocols are needed at each stage. In a multi-region mobile agent computing environment, places within a stage can be located in the same or different regions. The main advantage of this protocol is that this protocol reduces the overhead of stage works about two times as low as previous protocols so that it decreases the total execution time of mobile agents.

Now let’s describe some previously proposed techniques related with the security in mobile agent framework.

Hohl, Fritz [16] proposed code obfuscation technique to provide security to mobile agent execution. Code obfuscation is a technique which converts the readable agent code to machine or source code. There are many code obfuscator available online and offline. We can use them to convert our simple code in obfuscated code so that we can use it for security. For de-obfuscation of the process, we also have de-obfuscator available but there is no surety that obfuscated code is de-obfuscatable. This technique only provides execution privacy and that is low level security of mobile agent framework. Moreover it’s a preventive measure and its security directly depends upon computational capacity of agent execution environment.

Hyungjick Lee [15] stated the hybrid concept of functional composition and homomorphism technique (based on encryption scheme). This security mechanism worked on basis on encryption and decryption. The data communication in the system is fully based upon mobile cryptosystem technique. But the mobile agent’s access to private key is much needed at the execution platform which makes this technique less secure.

Ahmed [4] had explained a new security mechanism (SIM) to provide security to malicious hosts. In this technique, they proposed a technique in which clone or image of original agent is created and it is sent to next host for execution. After execution this agent comes back. There this clone or image agent is compared with original agent to find malicious changes. If some malicious changes are found then it means next agent is malicious and we can change the route of our original agent carrying data. But this process takes too much of time and sometimes follows unnecessary path in execution network.

Venkatesan, Chellappan [5] had proposed an approach using root canal algorithm. Which create the hash value of agent in bite code. Then it encrypts the value using its private key. Then code in encrypted and sent to remote server. Remote server or system gets the agent code with encrypted hash value. There remote system decrypts hashed code with the help of public key and find the alteration if occurs. In case, when malicious host changes the agent code then root canal algorithm is worthless. Moreover root canal algorithm have found worthless against Colluded truncation attack.

By analyzing above all previously proposed techniques related with fault tolerant and security of mobile agents we have found that in each technique if there is some advantageous, also there is something that is making it less effective and also these all proposed frameworks are fault tolerant or secure, but here we are going to propose a new integrated framework which is secure as well as secure against the malicious hosts.

3. Proposed Technique

The security in mobile agents can be classified into two catagories: *Agent security* and *data security*. In our approach we are going to deal with data security. We have two objectives for this paper: first , we have to make this agent framework fault tollerant for which we are going to use clone concept and checkpointing. Secondly, we have to make this framework secure from malicious attack using Blowfish encryption algorithm. We are going to use Blowfish encryption algorithm because of its reluctance to any attack and encryption time.

During our analysis of various encryption algorithms, we have found that Eliptic curve cryptography(ECC) and Blowfish Encryption algorithm are the unbeaten encryption algorithms. Though we have finalised Blowfish encryption algorithm because till now, No attack has been successful against this encryption standard.

3.1 Blowfish Encryption Algorithm

It is basically a symmetric block cipher having variable length key from 32 bits to 448 bits. It operates on block size 64 bits. It is a 16-round Feistel cipher and uses large key dependent S-Boxes. Each S-box contains 32 bits of data. Algorithm consists of S-Box and P-Box. The P-array consists of 18 sub-keys of 32-bit [17].

Blowfish has 16 rounds. The input is a 64-bit data element, x. Divide x into two 32-bit halves: xL, xR. Then,

```
for (i = 1 to 16)
        {
         $xL = xL \oplus Pi$
         $xR = F(xL) \oplus xR$
         Swap xL and xR
        }
```

After the sixteenth round, swap xL and xR again to undo the last swap.

Then, $xR = xR \oplus P17$ and $xL = xL \oplus P18$.

Finally, recombine xL and xR to get the cipher text.

Divide xL into four eight-bit quarters: a, b, c, and d. Then,

Function $F(xL) = ((S1,a + S2,b \bmod 232) \oplus S3,c) + S4,d \bmod 232$.

Decryption is exactly the same as encryption, except that P1, P2,..., P18 are used in the reverse order[7].

Above algorithm is the base of our security technique. Let’s create an e-scenario of an aglet network. Where one host is home agent server, which creates the agent and dispatches the agent to some node in an itinerary or Agent network.

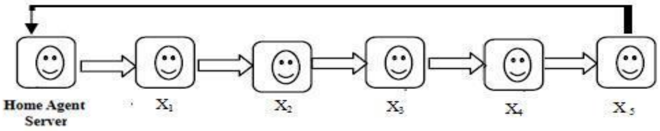

Figure 4. Agent Network

Our proposed Framework focuses on the fault tolerance and security of agent data. In order to achieve both in our fault tolerant and secure framework, we have used clone and check-pointing for fault tolerance and Blowfish encryption algorithm for encrypting data at home node.

3.2 Proposed Approach

Our proposed novel secure and fault tolerant technique make use of Blowfish encryption algorithm for data security of agent and clone, check-pointing for fault tolerant. So, to understand the working of this proposed technique we have to consider the following working diagram.

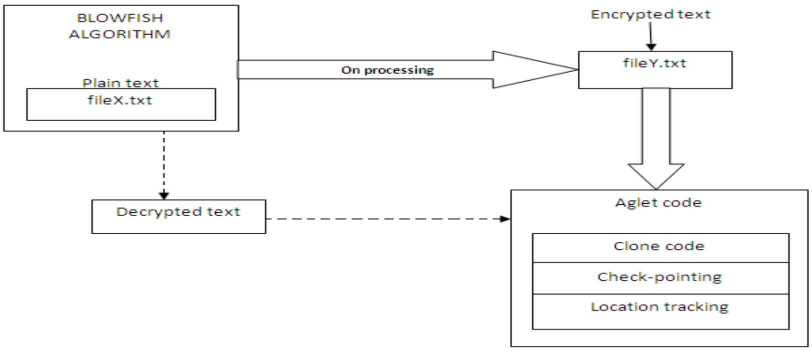

Figure 5. Proposed Technique Working Diagram

From above diagram, proposed technique can be understood clearly. When process starts, first home node owner generates the encrypted code or hash value of the plain text using the random key generated. This encrypted value or code is attached to mobile agent and dispatched to remote node or receiving node.

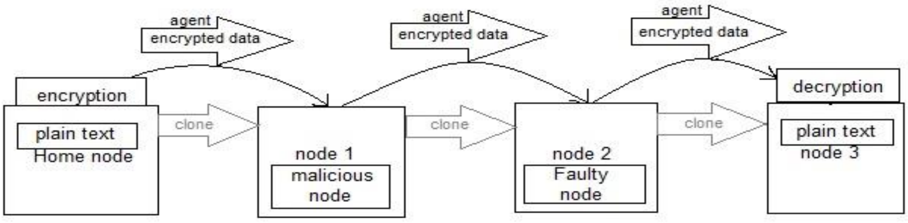

Figure 6. Agent Movement

Though, it is moving in an itinerary so there can be some malicious node or faulty node which can damage our confidential data or it can damage our moving agent. We are using clone and check pointing technique to avoid visiting faulty node. For security of data we are using blowfish encryption, which is very secure from malicious node. So if agent came across some malicious or faulty node, our proposed technique will make it able to bear this and agent will reach at desired remote location. Remote host will decrypt the encrypted code or hash value in plain text using random key generated by blowfish encryption algorithm. So using this technique we can safely move our data carrying agent over malicious and faulty environment.

3.2.1 Notation used in Our Proposed Technique:

M = Original Mobile agent
C = Mobile agent code
d_M = data part of mobile agent
D_M= digitally signed data
K_R = Random key generated
D_E = encrypted data
D_D = Decrypted data
C_C = Clone code
C_{CP} = check-pointing
Cl = Clone with check-pointing

We have divided the working of our proposed framework in two phases, one is *Home node phase* and another is *Remote node phase*:

- **Home node phase:**

- create mobile agent i.e. M(C)
- choose data part : d_M
- Applying digital signature i.e. $d_M \rightarrow D_M$
- Create random key K_R and encrypt text :
  $D_M \rightarrow K_R \rightarrow D_EK_R$(encrypted data)
- Encrypted text is attached with mobile agent code:
  M(C) D_E
- Encapsulating the whole code with clone and check pointing concept:
  $\{C_C \| C_{CP} \| M(C) \| D_EK_R\}$ + Cl (lags in time)
- This encapsulated code is sent to remote node.

- **Remote node phase:**

- Recipient host receives the code i.e.
  $\{C_C \| C_{CP} \| M(C) \| D_EK_R\}$ + Cl
- Encrypted text is recognized by remote host node and is retrieved from encapsulated code i.e. M(C) D_E
- Retrieve the digitally signed secret key and gave it to controller agent i.e. K_R
- Decrypts the encrypted value attached with the mobile agent code:
  M(C) $D_E \rightarrow K_R \rightarrow D_D$ (Original data)

This D_D is the decrypted value (plain text) that we got at the remote host. To check its integrity of this data can be checked by comparing:

M(C) D_E == M(C) D_D

Now we will analyze our proposed approach on the basis of different parameters:

3.3 Implemented Results Analysis

3.3.1 Encryption and Decryption Time: We will analyze encryption and decryption time by giving different sizes of text files to encryption algorithm as follows:

Table 1. Encryption/ Decryption Time of Different File Sizes

FILE SIZE (kb)	ENCRYPRION TIME (milliseconds)	DECRYPRION TIME (milliseconds)
822	313	39
1241	343	47
1669	444	63

2058	486	67

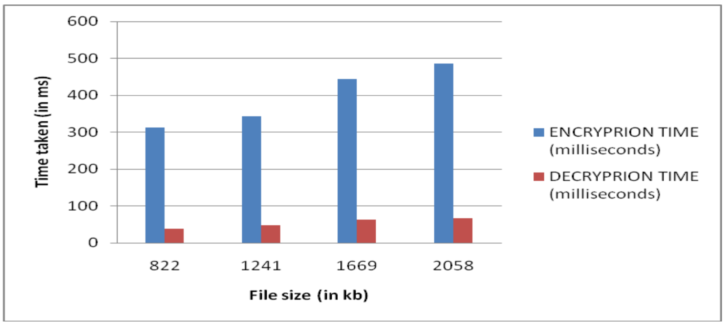

Figure 7. Execution Time of Encryption and Decryption

Above Bar-chart is showing the encryption and decryption time of the proposed encryption algorithm. Time taken for encryption and decryption can only be calculated in milliseconds because it takes very less time to encrypt and decrypt, even bigger size files.

3.3.2 Total Trip Time of Normal Agent and Secure & Fault Tolerant Agent: In this section, first we will calculate the total trip time of the normal agent (without fault tolerance and security). Secondly, we will calculate the total trip time of agent with security and fault tolerant code, *i.e.*, Fault tolerant and secure agent. Then we will compare their Total trip time and find the results.

Table 2. Total Trip Time of Normal Agent and Secure & Fault Tolerant Agent

Agent Location	TTOTAL TRIP TIME of Normal agent (in ms)	TTOTAL TRIP TIME of proposed agent (in ms)
node_1	0	0
node_2	47	63
node_3	95	111
node_4	142	158
node_5	189	205

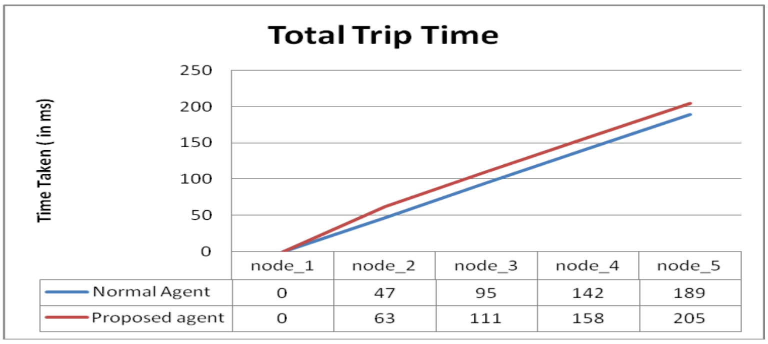

Figure 8. Comparison of Total Trip Time of Normal and Proposed Agent

3.3.3 Proposed Framework : Following table and Line diagram is showing the actual effect of our proposed technique in itenerery Vs Normal agent.

Table 3. Effect of our Proposed Technique on Working on an Agent

Agent Location	Normal Agent (at time x)	Proposed agent (at time x)	Clone agent (at time x)
Node_1	0	0	0

Copyright ⓒ 2015 SERSC

Node_2	47	63	65
Node_3	95	111	113
Node_4	Disposed	If agent (dispose)	160
Node_5			203

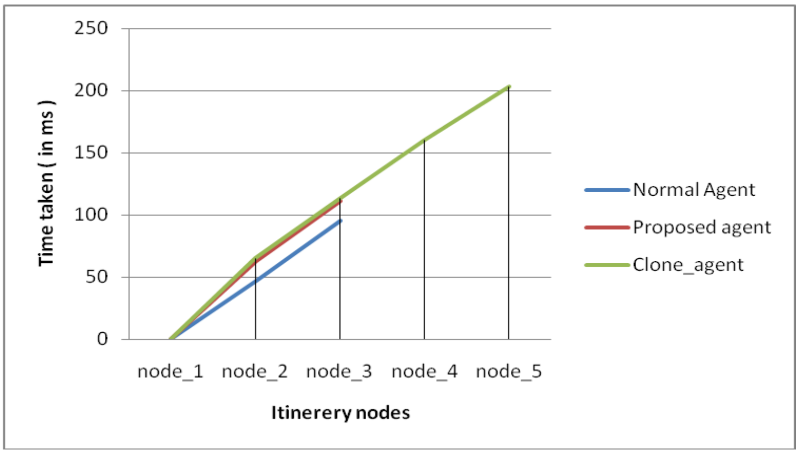

Figure 9. Line Chart Representation of Proposed Technique

In above Figure 9, Blue line represents the normal agent without fault tolerance and security technique. It moves from home agent but at third node it came across some faulty or malicious node and gets destroyed (or data amendment takes place). But in our proposed approach, our secure and fault tolerant agent moves from home agent, at third node it came across malicious or faulty node but in this technique agent will handle malicious node as it is carrying encrypted data. And if it is faulty node, there is also fault tolerance in this agent. Because its clone is also moving with original agent (clone also carries the encrypted data and fault tolerance code), so clone will recover the original agent if faulty node will destroy original agent. In line chart, maroon line is our original agent carrying data with it. If it gets affected by faulty node then Green line is showing that clone moving with original agent will recover and will complete the process designated to mobile agent.

From above graphs, it is clear that there is some overhead like time lag in our proposed approach as compared to normal agent, but it is negligible or bearable because it is in milliseconds (minor) and moreover our proposed framework or technique is more secure and fault tolerant.

4. Conclusions and Future Scope

In this paper we have described an integrated secure and fault tolerant technique which is better than the previously proposed techniques in terms of performance and security. This technique saves agent from faulty as well as malicious node. We have implemented the security by encrypting the agent data with Blowfish encryption algorithm, and fault tolerance with the help of clone concept and check-pointing. Agent carrying encrypted data moves with clone and clone performs the check-pointing process at specified nodes. Encrypted data is safe from any malicious node in the agent network. Thus agent safely reaches at desired node and decrypts the data to plain text. We have calculated encryption and decryption time. Also we have calculated Total Trip Time of normal agent and fault tolerant & secure agent in an itinerary. Our proposed technique is taking little more time than normal agent but it is providing integrated Security and Fault tolerance at same platform. In future, its Total Trip Time can be reduced by keeping agent size small.

References

[1] D. Chess, et al., “Itinerant Agents for Mobile Computing”, IEEE Personal Communications, vol. 2, no. 5, **(1995)** October, pp. 34-49.

[2] R. Pandey and N. Sharma, “Aglets (A java based Mobile agent) and its security issue” in International journal of emerging trends & technology in computer science (IJETTCS), vol. 2, Issue 4, **(2013)** July-August.

[3] S. Srivastava and G. C. Nandi, “Fragmentation based encryption approach for self-protected mobile agent”, Production and hosting by Elsevier B.V. on behalf of King Saud University, **(2013)**, pp. 1319-1578.

[4] T. M. Ahmed, “Using secure-image mechanism to protect mobile agent against malicious hosts”, World Academy of Science, Engineering and Technology, **(2009)**.

[5] S. Venkatesan and C. Chellappan, “Advanced mobile agent security models for code integrity and malicious availability check”, Journal of Network and Computer Applications, vol. 33, **(2010)**, pp. 661–71.

[6] L. Benachenhou and S. Pierre, “Protection of a mobile agent with a reference clone”, in Elsevier and science direct, computer communications, vol. 29, **(2006)**, pp. 268-278.

[7] J. W. Cornwell, “Blowfish Survey”, Department of Computer Science, Columbus State University, Columbus.

[8] F. Alan and D. Ralph, “Improving fault-tolerance by replicating agents”, In Proc. of the first international joint conference on Autonomous agents and multi agent systems, **(2002)**.

[9] S. Bagchi, K. Whisnant, Z. Kalbarczyk and R. K Iyer, “Chameleon: Adaptive Fault Tolerance Using Reliable, Mobile Agents,” In Proc. of 16th Symposium on Reliable Distributed Systems, ACM New York, NY, USA, **(1997)**.

[10] S. J. Choi, M. S Baik, H. S. Kim, J. W. Yoon, J. G Shon and C. S. Hwang, “Region-based Stage Construction Protocol for Fault tolerant Execution of Mobile Agent”, 18th International Conference on Advanced Information Networking and Applications, AINA, **(2004)**.

[11] A. Budi, I. Alexei and R. Alexander, “On using the CAMA framework for developing open mobile fault tolerant agent systems”, In Proc. of the international workshop on Software engineering for large-scale multi-agent systems, **(2006)** May 22-23, Shanghai, China.

[12] K. Damasceno, N. Cacho, A. Garcia, A. Romanovsky and C. LucenaIn, “Context-Aware Exception Handling in Mobile Agent Systems: The MoCA Case”, In Proc. of the 5rd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS 2006) at ICSE, **(2006)**.

[13] R. Hans and R. Kaur, “Novel Dynamic shadow approach for Fault tolerance in Mobile Agent system”, Goldcost 978-1-4673-2393-2/12/$31.00, IEEE, **(2012)**.

[14] Q. Zhang, Y. Mu, M. Zhang and R. H. Deng, “Secure Mobile Agents with Designated Hosts”, in IEEE Third International Conference on Network and System Security, **(2009)**.

[15] H. Lee and J. Alves-Foss, “The use of encrypted functions for mobile agent security”, In Proceedings of the 37th Hawaii International Conference on System Sciences, **(2004)**, pp. 1–10.

[16] F. Hohl, “Time limited black box security: protecting mobile agents from malicious hosts”, In Vigna, G. (Ed.), Mobile Agents and Security, Lecture Notes in Computer Science, Springer-Verlag, Berlin, vol. 1419, pp. 92–113, System based on JADE. Computers & Security, vol. 26, no. 3, **(1998)**, pp. 91–400.

[17] B. Schneier, “Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)”, [online] Available at: http://www.schneier.com/paper-blowfishfse.html.

Authors

Rajdeep Bhanot, Research Scholar, Computer Science & Engineering Department from DAV University Jalandhar, Punjab (INDIA). B. Tech. Professional with specialization in Information Technology from DAV Institute of Engg. & Technology (PTU) Jalandhar, Punjab (INDIA). Attended national level seminar on modern Cryptography techniques organized by PTU Jalandhar. He has some research publications in international journals.

Rahul Hans, He is working as assistant professor in the Department of computer Science and Engineering at DAV University, Jalandhar Punjab (INDIA). He has done his M. Tech in Computer Science and Engineering from Guru Nanak Dev University, Amritsar (INDIA). His research areas include Networking, Operating systems, Distributed computing, Mobile agents. He has various research publications in various international conferences and journal and also presented several papers in various national and international conferences.