Ilkogretim Online - Elementary Education Online, 2021; Vol 20 (Issue 1): pp. 6152-6176
http://ilkogretim-online.org
doi: 10.17051/ilkonline.2021.01.640

Analysis And Development Of Secure Management Of Volatile Information On Distributed System Environment Using Mobile Agent Paradigm

Rupal Chaudhary Research Scholar, Mewar University rupalchaudharymrt@gmail.com

Nirvikar Katiyar Director, Dr.Rizvi College of Engg. & Mgmt, Kaushambi

ABSTRACT
The most promising distributed system application paradigm contains the notion of a "mobile agent." A “mobile agent” is a software programme that travels through a “heterogeneous network” and operates independently at its destination. Mobile agent technology adds mobility to network communication by allowing mobile processes to relocate to new distant servers. Mobile agents can assist systems and real-time applications be more efficient and adaptable. Security issues plague “mobile agent” systems. Due to the “mobile agent paradigm” and threats, security measures must be established for both agent agents and hosting servers. There are various feasible solutions to the challenge of agent-hosting server security. Agent protection is a more complex technical challenge with fewer answers. The agent's owner does not control the behaviour of the agent-hosting servers, which adds to the complexity. This may distort the passing agent's calculations. Given the impossibility to force distant servers to comply with a security policy to prevent damaging the agent's data, the owner's cryptographic procedures may be a feasible solution. The “Broadcast Based Secure Mobile Agent Protocol” allows nonrepudiation, accountability, authorization, mutual authentication, secrecy, and integrity for distributed service applications. Attacks from the middle, repudiation, modification, and replay are also blocked. We verified the suggested protocol's efficacy using the Scyther verification tool.

Keywords: Volatile Information, Mobile Agent, heterogeneous network, mobile agent paradigm.

I. INTRODUCTION

Mobile agents are self-contained software programs that may roam between servers via a network, conducting calculations on the user’s behalf. For a sizable class of “network-centric applications,” the “mobile agent paradigm” offers an appealing alternative to traditional “client-server” development. It may be utilized in various applications, giving advantages such as reduced bandwidth use and customized clients and servers. Thus, security is the primary impediment to the widespread deployment of “mobile agents.” We have examined possible dangers to “mobile agents” in this section and some presently accessible techniques for mitigating the risks associated with this technology.

Due to the unique nature of mobile agent data, static and dynamic data components should be safeguarded independently. While basic cryptographic algorithms may be used to secure static data, safeguarding dynamic data presents a greater challenge. There is a dearth of research devoted to protecting dynamic data in a mobile agent. We investigated many published strategies for maintaining the integrity and privacy of an agent's dynamic data.
However, they are susceptible to certain assaults or do not adequately safeguard this data. Additionally, we discovered no architecture for embedding security procedures into a solution for "mobile agents." The objective is to develop a hybrid solution that uses asymmetrical and symmetrical encryption. The system can meet a variety of security needs that are more difficult to meet by utilizing asymmetrical cryptography, such as symmetrical encryption, mutual authentication, nonrepudiation, and accountability, by employing asymmetrical cryptography.
Special purpose agents may be conceived of as a group of other agents working together to offer services and a computational environment for other agents. Like an operating system kernel, these trustworthy processes allow user processes to run by providing fundamental duties such as scheduling and memory management. If a service is invoked (i.e., a trusted agent within the platform is called upon), it captures and packs the state information of the calling agent, serializes the code of the calling agent, and dispatch the archived representation to the specified destination. An agent platform may be discussed and organized like this approach, but this does not imply that an agent platform must be executed in that manner.
An "intelligent software agent" is a program or module that works on behalf of the user. "As a result, software agents may be characterized as intelligent software modules that users assign to do tasks on their behalf. Remote Method Invocation (RMI) and mobile agent technology are two of the most used methods for sending messages in a distributed setting. We looked at many different metrics for this comparison, including invocation costs, fault tolerance, and search time. Experimentation shows that "mobile agents" outperform Remote Method Invocation (RMI) in terms of performance. Invocation costs for "mobile agents" are cheaper, and calculation times are quicker. Furthermore, it is claimed that systems using "mobile agents" have a 10% failure rate, resulting in a more gentle decline and, as a result, greater fault tolerance.

A. MOTIVATION

"Mobile agents" are software entities that deal with many systems in a distributed scenario. Hence, it is inappropriate to utilize them without proper precautions and mitigation mechanisms. The precautions and the mitigation measures vary with the type of application and completely depend on the type of "mobile agent" used in the system. It also depends on the threat over the application. Security is not an optional factor in applications but is a vital one. Though there are several security mechanisms, it may not be possible to overcome all the threats posed by an application. However, the risk of the attacks can be minimized, or the system can be subjected to mitigation even if an incident occurs.

B. AIM OF THE STUDY

The “mobile agent” carries data from one platform to another. Therefore, security for the data carried by the “mobile agent” and the design of a decision support system that helps the “mobile agent” to decide its travel plan plays a key role.

C. CONTRIBUTIONS AND ACCOMPLISHMENTS IN THE FIELD

The following are the most significant results:

- Identification and classification of vulnerabilities in mobile agents
- The establishment of a new security property
- The definition of criteria for constructing schemes that maintain integrity and privacy.
- The proposal of two novel security mechanisms, including “recoverable key commitment” and “conditionally anonymous digital signatures”.
- Development of new security systems and technologies
- The design of the new security architecture is step number six.

II. LITERATURE-REVIEW

To accomplish an objective, the “mobile agents” may move between different hosts. During the itinerary, mobile agents face several threats when they are subjected to move through multiple hosts, which may impact the state and data carried by them. Initially, mobile agents were considered threatening because most people feared affecting their systems. The vice versa was also equivalently true that the mobile agents are affected by the hosts.

A. DATA SECURITY

Yao et al. (2013) identified a “colluding servers” attack, where two servers cooperate to conspire against a mobile agent and modify the data carried by it. Forward Integrity is the most important security property which a mobile agent is expected to satisfy (Karjoth et al., 2008).

Zhou et al. (2004) clearly showed that the methods might suffer from colluded truncation attacks. They also proposed a scheme that provides data confidentiality, forward privacy, forward integrity, insertion defense, and truncation defense (Zhou et al., 2004a). The proposed scheme overcomes the result-truncation attack.

To avoid two collusion attacks, Songsiri (2005) presented a chain signature verification approach. The strategy made it easier to detect different kinds of collusion attacks and to identify hostile sites that were involved in them.

“One hop backward and two hops ahead” chain relation was suggested by Xu et al. (2006) as a better free-roaming mobile agent security protocol to combat collusion truncation attacks.

Shao and Zhou (2006) proposed two schemes to overcome the blocking attacks by malicious hosts over mobile agents. The schemes were simple schemes and <t,n> fault-tolerant schemes. The results prove that the <t,n> tolerant scheme is better against blocking attacks.

Xian and Feng (2007) proposed a private key consignment method to protect mobile agents. This method used a trusted computing methodology that uses both public key and symmetric key cryptography to protect the data and key. It also used tamper-proof hardware to protect the private key.

Garrigues et al. (2008) proposed a mechanism based on trust to protect dynamic itineraries of mobile agents, while most security mechanisms focussed on protecting static itineraries. The mechanism addressed the issues, namely tampering (alteration), impersonation (masquerading), and disclosure (threat to confidentiality).
Security researchers Linna and Jun (2010) suggested an encryption approach to defending mobile agents from colluding truncation attacks by encrypting the data into its divisible whole. When the "mobile agent" reaches its location, it encrypts and delivers the identification information it has gathered to a third-party trusted by the agent. In the event of a suspected assault on the data, the agent will report back to the trusted third party.

B. CODE AND AGENT SECURITY

In Zhong et al. (2004), a connection migration mechanism was developed to ensure that all data communicated at the moment of data migration was delivered at the same time. It incorporates an access control mechanism to control the network ports the mobile agents get transmitted. To avoid multiple authentication requests, the session is maintained concerning each connection.
Jiang et al. (2004) deploy the “agent migration fault tolerance model-based” on the “integrity verification method” to achieve fault-tolerance and integrity verification in mobile agents. This method provides a fault-tolerant system and overcomes the attacks faced by mobile agents during the itineraries.
Xu (2004) used a collection of cryptographic protocols to provide security to “mobile agents”. The major objective was to remove the dependency on a trusted third party. The mobile agent must be aware of the protocol deployed over the framework to achieve this. The system was a fault-tolerant and secure one.
In most of the mobile agent security mechanisms, the “mobile agent platform” is responsible for the security of the mobile agent. In contrast, Ametller et al. (2004) proposed a system where agents protect the code and data by their security mechanism. The major advantage is that this mechanism can be implemented in any system deploying mobile agents without impacting the existing codebase. It provides a framework that any development platform can inherit.
Hacini et al. (2006) proposed mobile agent architecture based on the dynamic adaptability of “mobile agents”. The behavioral aspect of the mobile agents’ pattern in this system cannot be studied. This reduced the probability of the mobile agent being attacked by malicious hosts.
Wu et al. (2006) harvested the benefits of trusted computing to protect mobile agents. The trusted platform module was integrated with the mobile agent platform to provide authentication, integrity services, secure group communication, and storage, ensuring mobile agents’ security.
Kumari et al. (2007) proposed a policy-based framework for migrating mobile agents to unknown hosts or platforms to collect information that provided good trustworthiness and security during their itinerary.
Ou et al. (2008) proposed a privilege management infrastructure based on an “X.509 attribute certificate-based access control” mechanism to prevent unauthorized access of malicious entities over agent systems. The major advantage of this system is that the access control system is dynamic.

Hong (2009) suggested an efficient, secure (t, n) threshold proxy signing approach based on the RSA cryptosystem. In this method, the proxy signature generating process is independent of the proxy signature combining step. The system is also independent of the number of proxy signers.
Garrigues et al. (2010) proposed a "software architecture" and development environment to construct an application employing secure mobile agents. The design offered a collection of lightweight cryptographic protocols and permitted the reuse of protocols by mobile agents. This tremendously boosted the development of secure apps employing mobile agents.
Marikkannu et al. (2011) identified tailgating attacks, a new type of attack in mobile agents. Dual-checkpoint analysis was proposed to overcome the tailgating attacks in mobile agents. This method focuses only on verifying the integrity of the mobile agents and does not address any other security issues relevant to mobile agents.
Srivastava and Nandi (2014) built a protocol having integrity-based confidentiality and a self-protection-based approach. For security reasons, the mobile agents are made less interactive during the execution phase. The protocol was implemented using JADE and was checked using Petri's net-based formal representation of the protocol.
Zrari et al. (2015) proposed a system to protect stationary agents when communicating or interacting with a mobile agent. Hedin and Moradian (2015) presented a model for securing agents systems using the role and communication exhibited by mobile agents.
Bagga et al. (2017) proposed the "self-adaptive IV-Phase MAP security architecture." This is a biological immune system-based approach for detecting malicious mobile agents. Various data mining methods are used in this framework to identify malicious agents. The architecture is efficient against "man-in-the-middle" attacks, masquerading, replay, repudiation, and unauthorized access attacks. This framework exhibits maximum machine learning algorithms to secure a mobile agent system.

C. POLICY-BASED SECURITY

Jansen (2016) proposed a system for controlling the behavior of "mobile agents" based on certain rules. This system contains an access control list which provides a detailed list of resources accessible by a mobile agent. This control list is protected utilizing digital signatures. This system controls the access of mobile agents towards a resource, thereby protecting the platform and data contained by the platform.
Zou and Liu (2017) presented a model based on an opinion-based belief structure for mobile agents to select the most suitable host. The opinion-based belief structure calculates the host's reputation and ranks the host. Thus this model acts as a recommender system for mobile agents.

D. AGENT RECOVERY

Marikkannu et al. (2011b) used the concept of adaptive mobile agents to overcome failures in mobile agent systems. A conclave environment is proposed inside which the mobile agents execute in this system. At the time of execution, if a mobile agent is subjected to failure, a mobile agent who has completed its task can adapt to the role of the mobile agent which had failed. By this means, the system achieves fault tolerance.

E. OVERVIEW OF SECURING

Many researchers have looked at the issue of security. The privacy of system users and their ability to remain anonymous are two issues that several suggested solutions attempt to solve (Burkle, A et al. 2006, CentOS 2013). This capability is essential for applications that deal with sensitive data. (Ftima, FB, Tounsi, W, Karoui, K &Ghezala, HB 2009). Writers have also proposed Secure MASs using cryptographic or trustworthy platforms (Garrigues, C et al. 2008, Garrigues, C 2010, Garrigues Olivella, C et al. 2008, Hacini, S, et al. 2008, Hedin, Y et al. 2015).

When it comes to anonymity and secrecy for users, (CentOS 2013) is a good place to start. The MAs stop at mixers, intermediate nodes that serve as intermediary nodes during their journeys. The addresses of mixers visited throughout the MA route are stored in the agent state. The MA continues to roam about, randomly moving from node to node. At the end of the journey, the addresses of the mixers visited in Massachusetts are utilized to lead the agent to the platform owned by the mixer's owner. The protocol shields the owner of the MA and the route taken by the agent. Encryption ensures that only those who have the proper permissions may access data. Users are not authenticated, and it is presumed that all communications sent are from genuine users, as no authentication is supplied.

Using the notion of a last input first output (LIFO) buffer of data, the authors (Chuanrong, Z &Yuqing, Z 2009) suggested a new anonymity approach. Secret key, hash, nonce generated by each platform is unique and cannot be used on any other platform. A secret key is used to store the nonce and hash in the stack, along with the previous destination, after decryption. After the agent completes its journey, each platform uses its secret key to decrypt the information contained on the stack and return the agent to where it began. The route saved is checked to determine whether the stack has been tampered with using the saved hash.

They proposed a cryptographic and trustworthy platform-based protocol that assures the secrecy of the dynamic itinerary that is established at run time (Chung, T.S., Chen & Lai, M.W. 2009). Platforms that collect data from other platforms and arrange an agent's next journey are the most essential component here. To guarantee that only the specified platform has access to the itinerary information, the information has been encrypted. Agent code is protected from tampering by the protocol's hash and asymmetric encryption in addition to the privacy it offers for agents' travel plans. It is made possible for mutual authentication between MAs and platforms. There is a record of the number of executed agents on each platform. An agent must first verify that the trip marker is already in the platform's database before he or she can proceed. If there is no match, the agent information is stored and the program is launched. To determine whether or not a replay has occurred, the counter's value on the agent and the value it has in its database are examined. If the counter has been incremented, it's considered lawful, and if it has not, it is destroyed. The protocol is tested using the Java Agent Development Environment (JADE) to simulate replay attacks (Bedi, P et al. 2008, Beheshti, S et al., 2007). Results show that the proposed protocol can detect all replay attempts, although it is still susceptible to additional attacks like eavesdropping and repudiation. Because each copy of the agent has to have a unique trip identifier, it is not supported by the proposed protocol for mutual authentication and authorization. In addition, the MAS's acquired data is lost when the MAS is maliciously destroyed (Hijazi, A & Nasser, N 2005).

Agent information, such as its owner's name, timestamp, certificate, and other necessary data, is given to the agent in the form of a passport. Authentication servers (ASs) verify the passport and provide a

visa containing all the MA's resources. Platforms can tell whether a malicious agent attempts to access restricted resources using a Visa issued by a reputable Visa and an authentication server. The process ensures that the passport and visa are authentic. Secure communication channels are assumed to be available. Passport and visa information received in plain text may be accessed if the channel is hacked. A denial of service attack might occur if attackers intentionally tamper with an agent's passport and visa and transmit their altered version to genuine overload platforms, even though the passport and visa may not include important information (Hong, X 2009).
This protocol is an upgrade. Code signatures signed by XRC give evidence for identifying bad platforms, and hence protection against repudiation attacks. e Xtended Root Canal (XRC) (Hedin, Y et al. 2015).
A security architecture for mobile agent systems was presented (Jansen, WA 2011). Multi-level access control of platform resources for MAs is introduced in the proposed framework. The degree of access an agent has is determined by the number of times they've walked the same route. The MA checks off each station it visits before deciding where to go next. Platforms look up the history of previously visited platforms to determine how trustworthy the MA is and then access it depending on that assessment. One of the drawbacks is that the platform can recognize hostile platforms in the system.
It has been suggested that MASs use a matrix hop architecture (Ftima, FB, Tounsi, W, Karoui, K & Ghezala, HB 2009). MAs are protected by the system's design against being maliciously killed or lost. Before the transfer of the MA, all data created on each platform is returned to its owner platform. As a result, the design allows the system to recover even if the agent is destroyed. As long as the owner platform maintains track of the data given by other platforms, the owner platform will transmit it to the next destination platform to continue its trip if the agent is destroyed.
E-health applications may benefit from using a secure MA data transmission mechanism (Garrigues, C, Robles, S &Borrell, J 2008). The decryption information is only made available when the platform has been verified to be legitimate. The first step is to provide the owner's agent with a token and encrypted data for verification. The token is sent back to the sender when the receiver verifies the agent's identity and the integrity of the code and provides further evidence of the token's authenticity. The nonrepudiation of data, the integrity of Mobile Agent data, and confidentiality of the MA's journey are all addressed in (C, Borrell, J, Robles, S, Garrigues, & Navarro-Arribas, G 2010).
The Knowledge-Based System (KBS) in (Garrigues Olivella et al. 2018) gathers information regarding the trustworthiness of hosts. Trusted hosts and chain relations are the foundation of the protocol. Initiated by the owner host and directed by the KBS, the MA then moves on to its next location. The MA visits hosts to gather information. Data acquired by the MA must be verified by the trust hosts when they visit. It is notified to the owner host whenever there is any change in the data. Data secrecy, non-repudiation, and integrity are provided through asymmetric encryptions and hash functions. As a result, only the MA owner and trusted hosts have access to the owner's identity and the route traveled by the MA.
It is suggested to use symmetric cryptography (Hedin, Y & Moradian, E, 2015). Symmetric keys can only be generated and distributed in this manner. Keys are obtained from platform-generated data during runtime. Since it is more difficult for attackers to guess or anticipate the keys, it is more difficult. Confidentiality, data integrity, and authentication for the agent owner are all given.

III. RESEARCH METHODOLOGY

In addition to providing basic security needs, BBSMAP protects against various threats. Even if one of the MAs is deliberately destroyed or lost, the proposed broadcast design ensures that the system will continue to work and perform well simultaneously.

A. PROBLEM STATEMENT

"What are the various security requirements of Volatile Information in Distributed system environment? How to securely manage volatile information using a mobile agent?"

B. THE GOAL OF THE RESEARCH

This project aims to develop a solution that uses both symmetrical and asymmetrical encryption techniques. The use of asymmetrical cryptography allows the system to meet a wide range of security needs that are difficult to meet with symmetrical encryption, such as mutual authentication, accountability, and nonrepudiation, without compromising performance.

C. OBJECTIVE OF RESEARCH

In our study, we have the following two objectives:

- To study the various security requirements of Volatile Information in the Distributed system environment.
- To find a securely manage volatile information using a mobile agent.

D. SCOPE OF THE RESEARCH

The scope or area of concentration you choose as a researcher is critical. Do not forget, though, that expanding the scope of your project too much might result in you being unable to accomplish it, or it could take a long time. Before you lay down the details of your project, think about whether or not it's doable. The results may not be generalizable if the scope is too small. By applying our proposed method, we can assure the security of volatile information in a distributed system environment using Mobile agents.

E. PROPOSED BBSMAP

It is used in BBSMAP to take a hybrid approach to cryptography, which involves employing both symmetrical and asymmetrical cryptography. The use of asymmetrical cryptography allows the system to meet a wide range of security needs that are difficult to meet with symmetrical encryption, such as mutual authentication, accountability, and nonrepudiation, without compromising performance. The use of symmetrical cryptography, on the other hand, improves the performance of the system since it is less demanding in terms of operating resources. As a result, implementing the hybrid method ensures that the system meets all necessary security standards while also providing excellent performance.

Notations and definitions are supplied in (I) in the "Notations" section to offer a more detailed explanation of how the interaction is carried out. Fig. 3.1 is an example of how the proposed BBSMAP for distributed applications may be used in a real-world setting. As an example of a distrusted application, a scenario involving a location-based service (LBS) is shown. After being validated by the

telecom operator, service providers are approved for registration. Service providers identify themselves, disclose their physical address, and describe the services they provide. It is also possible that their reputation will be a significant element in the registration procedure. We also assume that public key infrastructure (PKI) has been established.
Our scenario involves service providers providing customers with names of potential hotels, restaurants, and airline agencies located in the vicinity of the requested location. Various alternatives are shown to users, each with more information about the option, such as the average price and distance from the user's current location. The information offered assists users in making decisions throughout the decision-making process.

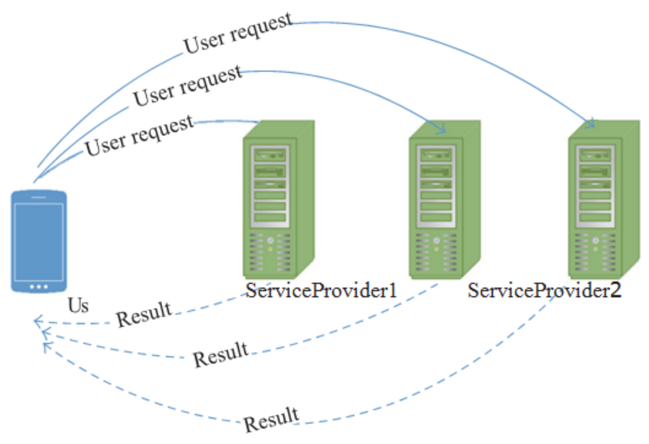


Fig. 3.1: System scenario for BBSMAP as implemented in a distributed application

The user submits their request after gaining access to the program. The request is then published to all of the registered service providers. Upon receiving the request, each service provider examines its database for matching results, then delivers it back to the customer. It is still possible to get results from the other service providers even if one of the broadcast messages is lost or misdirected. The following is an example of a user request: "Please give me the names of nearby hotels," to which a potential response maybe "The Sheraton Hotel is 10 miles away and costs on average $100 per night."

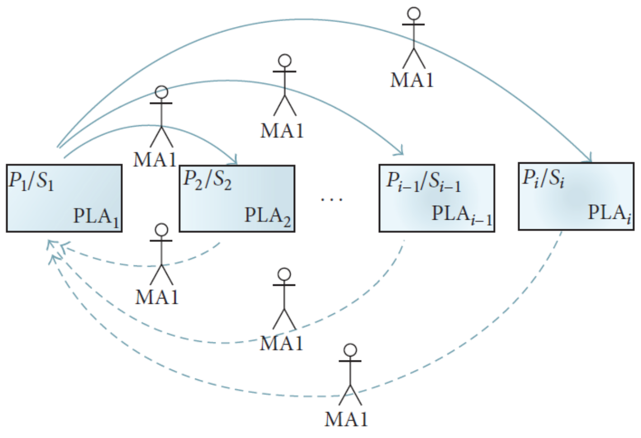


Fig. 3.2: The design of the BBSMAP system

The architecture of the BBSMAP system is seen in Fig. 3.2. PLA1 is the platform that owns the MA and is responsible for launching copies of the MA to service providers, including the user's request, including the location and service type. As shown, the system contains i1 service providers platforms, which range from PLA2 to PLAi in terms of size. MA1 is generated by the platform PLA1, which is the owner platform. It is the responsibility of MA1 to transport the user request R to gather the output of requests ORi from platforms, which contains various service alternatives for users.
The specific phases of the interaction between the owner platform and service providers are described in detail.

F. INTERACTION BETWEEN OWNER AND SERVICE PROVIDER PLATFORMS

For the sake of simplicity, here, we do not discuss certificates distribution as we assume that platforms have been verified and granted certificates by the certificate authority (CA) (Kocarev, L et al. 2011). Stages of interaction between owner platform PLA1 and PLAn are described. The PLAn is any service provider in the system where i and n are positive integers and $2 \leq n \leq i$:

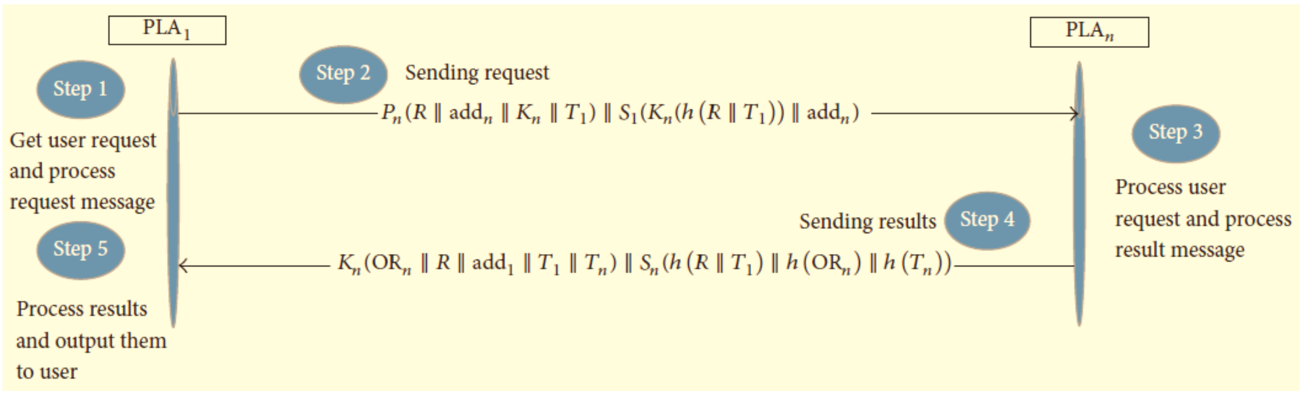

PLA$_1$
PLA$_n$
Step 1
Get user request and process request message
Step 2
Sending request
$P_n(R \parallel \text{add}_n \parallel K_n \parallel T_1) \parallel S_1(K_n(h(R \parallel T_1)) \parallel \text{add}_n)$
Step 3
Process user request and process result message
Sending results
Step 4
$K_n(\text{OR}_n \parallel R \parallel \text{add}_1 \parallel T_1 \parallel T_n) \parallel S_n(h(R \parallel T_1) \parallel h(\text{OR}_n) \parallel h(T_n))$
Step 5
Process results and output them to user

Fig. 3.3: Process of exchange of message b/w PLA$_1$ and PLA$_i$

G. SECURITY ANALYSIS OF BBSMAP

The following section contains an examination of BBSMAP's security measures. We demonstrate that BBSMAP satisfies the following security specifications.
Communication entities must authenticate one another's identities for communication to be successful. Take, for example, the interaction between PLA1 and PLAn. This is a complex interaction. Both systems can identify one another via the use of public key infrastructure (PKI) signatures and certificates. Additionally, they are confident in who sent the communication because of the digital signature. By using timestamps, both platforms can verify that the messages exchanged are indeed meant for them, and by concatenating the address of the destination in the message, both platforms can verify that the messages sent are intended for them. Furthermore, communications are received properly, and PLAn responds to PLA1 as soon as it gets the request. Since the interaction meets all of the conditions for mutual authentication, BBSMAP can provide mutual authentication.

Accountability: BBSMAP provides accountability in the sense that if platform PLA2 decides to act maliciously and change the request, then PLA1 detects the change by comparing the hashes, and the PLA2 signature serves as proof of its malicious action.

Security: BBSMAP provides security in the sense that if platform PLA2 decides to act maliciously and change the request, then PLA1 detects the change by comparing the hashes. Since our protocol is being suggested for distributed services applications, our proposed architecture comprises both consumers (senders) and service providers (servers).
In this step, the user visits the application and enters their credentials. This information is then checked to determine whether or not the user should be granted access to the program's services. As a result, only approved platforms (service providers) can give the service to the user since the certificate authority examines the authenticity of the certificate and their own identity.

Modification attacks: Because of the digital signature and the integrity checks that compare hash values at each platform, a hostile platform or agent cannot alter the timestamps, the request, or the results without being noticed.

Confidentiality: Because of public and symmetric key encryptions, the information shared between platforms stays private during the conversation. Because the output of the request is encrypted with the secret session key, only the platform that issued the request will be able to access it at any time.

Replay attacks: the time stamp shows the freshness of the message, and any message that has passed its expiration date is rejected. Both the accuracy of the data and the timeliness of delivery are reviewed. There are two timestamps in use. Two timestamps are generated: one is used to mark the time of the creation of the request, and another is used to stamp the time of the creation of the data (results). The user will only consider the given results if both timestamps are current; both the request and the produced data are current and fresh.

Nonrepudiation: The signature of platforms serves as confirmation that the intended recipient delivered the message. When using BBSMAP, platforms sign messages with their private keys, allowing for the potential of action traceability to be established.
A hostile platform cannot imitate another platform due to digital signatures, which prevents masquerade attacks from occurring. Scyther is used to carry out formal verification proofs to ensure that BBSMAP is correctly implemented. Following that, a thorough description of the verification tests and outcomes is provided, and a comparison of BBSMAP with other current protocols in the field.

H. FORMAL VERIFICATION AND VALIDATION

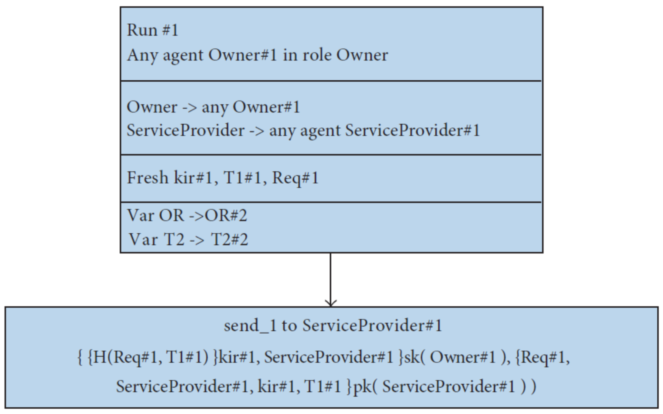
We choose the Scyther formal methods verification tool to test BBSMAP's functionality. Scyther is a formal analysis tool that may validate security protocols. It is completely free. Scyther presents a collection of claims that may be used to test the secrecy of information, the synchronization of data, and the authentication of persons involved in communication. In communication, synchronization refers to the fact that the intended communication party receives messages sent and that messages received are transmitted. It also implies that the communications transmitted and received have not been tampered with and that their contents have not been altered. Scyther also checks for the

aliveness of a communication party, which simply implies that the party is still alive and has carried out some actions since it was last checked (Kumari, VV et al. 2007, Kun, G et al. 2010).
Fig. 3.4 depicts the verification test results performed on the specified claims Secrecy is checked by comparing the output of request, request, the transferred time stamps, session keys, and private key to a list of known values. Furthermore, the liveliness of the parties and their synchronization with one another are checked.
To depict the interaction and messages transferred between communication partners, Scyther creates trace figures (Fig. 3.5 – 3.7). The interaction between BBSMAP and its environment may be divided into three phases. Trace diagrams are used to illustrate each step of the process. Because Scyther employs its notations that vary from those previously supplied, sections (II) of the "Notations and Definitions" section contain a list of the new notations used in Scyther traces.
The first step, which is to transmit the user request to the service provider platform, is explained in detail in Fig. 3.5. As seen in the diagram, the message is sent from the owner platform (Owner#1) to the service provider platform (ServiceProvider#1) through the service provider platform (ServiceProvider#1).
Fig. 3.6 depicts the second step of the process. ServiceProvider#1 is responsible for processing the request message submitted by Owner#1. Using the match functions, ServiceProvider#1 authenticates Owner#1 and verifies the integrity of the information that has been delivered. In a logical comparison, match functions compare two values to determine whether or not they are equal. By comparing the hash sent in the message with the hash of the newly received data produced on the present platform, match functions are used to ensure the integrity of the data and time stamps transmitted and received. ServiceProvider#1 creates and transmits back to Owner#1 the integrity test results passed by the system.

```
usertype TimeStamp,Data,Request,Key;
hashfunction H;
macro hash1=H(Req,T1);                        % hash the request and time stamp 1
macro hashofdata=H(OR);                       % hash the result
macro hashoftime=H(T2);                       % hash the request and time stamp 2
protocol MyProtocol(Owner, ServiceProvider){
role Owner { fresh kir:Key;                   % generate session key
fresh T1:TimeStamp;                           % generate time stamp 1
var T2:TimeStamp;                             % prepare variable to receive time stamp 2
const Req:Request;                            % generate request
var OR:Data; % prepare variable to receive result
send_1(Owner, ServiceProvider,
{{hash1}kir, ServiceProvider}sk(Owner),{Req, ServiceProvider, kir, T1}pk(ServiceProvider));
recv_10(ServiceProvider, Owner, {hash1, hashofdata, hashoftime}sk(ServiceProvider),
{OR, Req, Owner, T1, T2}kir);
macrohashofdata2 = H(OR);                     % generate hash of received result
macrohashoftime2 = H(T2);                     % generate hash of received timestamp
match(hash1, hash2);                          % compare hashes
match(hashofdata, hashofdata2);
match(hashoftime, hashoftime2);
claim_i1(Owner, Nisynch); % test Claims
claim_i2(Owner, Alive);
claim_i3(Owner, Secret, OR);
claim_i4(Owner, Secret, T1);
claim_i5(Owner, Secret, T2);
claim_i6(Owner, Secret, Req);
claim_i7(Owner, Secret, Key);
claim_i8(Owner, Secret, sk);
}
role ServiceProvider{
fresh OR:Data;                          % generate result
var kir:Key;                            % prepare variable to receive session key
 fresh T2:TimeStamp;                     % generate time stamp 2
 var T1:TimeStamp;                       % prepare variable to receive time stamp 1
var Req:Request;                        % prepare variable to receive request
recv_1(Owner, ServiceProvider,
{{hash1}kir, ServiceProvider}sk(Owner), {Req, ServiceProvider, kir, T1}pk(ServiceProvider);
macro h2=H(Req,T1);                      % generate hash of received request and time stamp 1
match (hash1,h2);                       % compare hashes
send_10(ServiceProvider, Owner,
{hash1, hashofdata, hashoftime}sk(ServiceProvider), {OR, Req, Owner, T1, T2}kir);
claim_i9(ServiceProvider, Nisynch);              % test claims
claim_i10(ServiceProvider, Alive);
claim_i11(ServiceProvider, Secret, OR);
claim_i12(ServiceProvider, Secret, T1);
claim_i13(ServiceProvider, Secret, T2);
claim_i14(ServiceProvider, Secret, Req);
claim_i15(ServiceProvider, Secret, Key);
claim_i16(ServiceProvider, Secret, sk);
}}
```

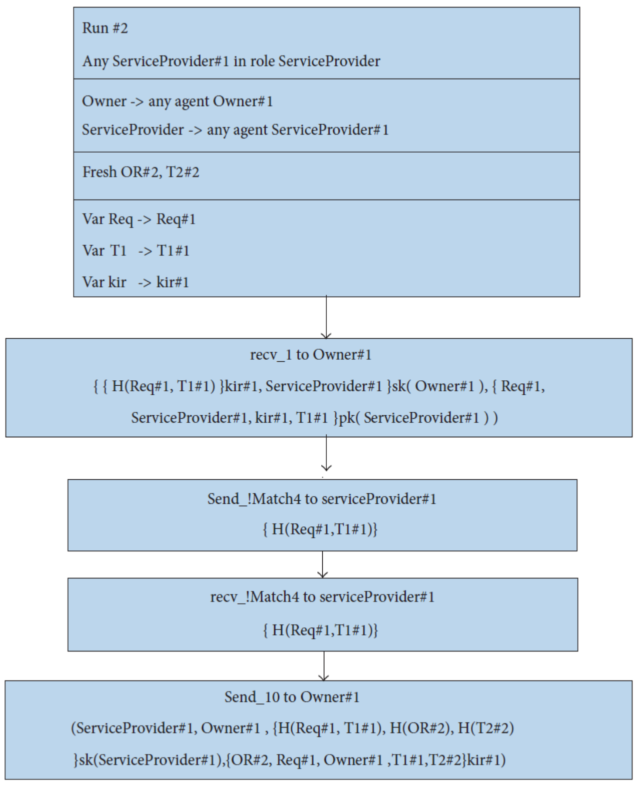
Algorithm - Verification Code

The verification test results conducted on the stated claims are shown in Fig. 3.4. Figure 3.4 A list of known values is used to verify secrecy by comparing the transmitted time stamps, the request, the output of the request, the private keys, and the session key to a list of known values. Aside from that, the parties' liveliness and synchronization are also monitored closely.

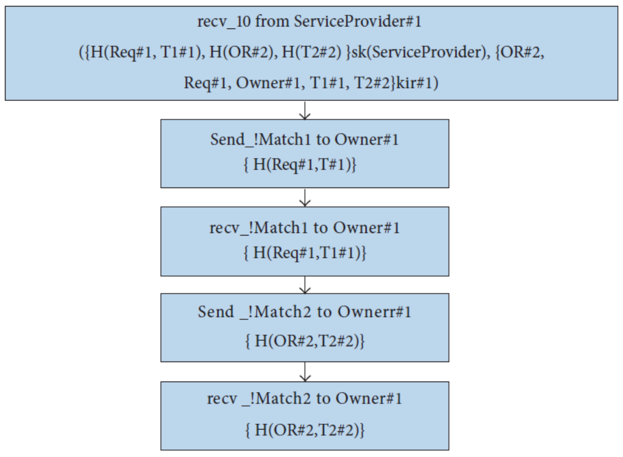


Run #1
Any agent Owner#1 in role Owner
Owner -> any Owner#1
ServiceProvider -> any agent ServiceProvider#1
Fresh kir#1, T1#1, Req#1
Var OR ->OR#2
Var T2 -> T2#2
send_1 to ServiceProvider#1
{ {H(Req#1, T1#1) }kir#1, ServiceProvider#1 }sk(Owner#1), {Req#1, ServiceProvider#1, kir#1, T1#1 }pk(ServiceProvider#1))

"Fig. 3.5: Step 1- Owner Owner#1 sending request to Service Provider ServiceProvider#1."


Run #2
Any ServiceProvider#1 in role ServiceProvider
Owner -> any agent Owner#1
ServiceProvider -> any agent ServiceProvider#1
Fresh OR#2, T2#2
Var Req -> Req#1
Var T1 -> T1#1
Var kir -> kir#1
recv_1 to Owner#1
{ { H(Req#1, T1#1) }kir#1, ServiceProvider#1 }sk(Owner#1), { Req#1, ServiceProvider#1, kir#1, T1#1 }pk(ServiceProvider#1))
Send_!Match4 to serviceProvider#1
{ H(Req#1,T1#1)}
recv_!Match4 to serviceProvider#1
{ H(Req#1,T1#1)}
Send_10 to Owner#1
(ServiceProvider#1, Owner#1 , {H(Req#1, T1#1), H(OR#2), H(T2#2) }sk(ServiceProvider#1),{OR#2, Req#1, Owner#1 ,T1#1,T2#2}kir#1)

“Fig. 3.6: Step 2 – the service provider ServiceProvider#1 is now processing the received user request and sends the result back to the owner Owner#1.”


recv_10 from ServiceProvider#1
({H(Req#1, T1#1), H(OR#2), H(T2#2) }sk(ServiceProvider), {OR#2, Req#1, Owner#1, T1#1, T2#2}kir#1)
Send_!Match1 to Owner#1
{ H(Req#1,T#1)}
recv_!Match1 to Owner#1
{ H(Req#1,T1#1)}
Send _!Match2 to Ownerr#1
{ H(OR#2,T2#2)}
recv _!Match2 to Owner#1
{ H(OR#2,T2#2)}

“Fig. 3.7: Step 3 – the owner Owner#1 processing the result received from the service provider ServiceProvider#1. “

Fig. 3.5 explains the first stage, which is the broadcast of the user request to the service provider platform. The figure shows that the owner platform (Owner#1) sends the message to the service provider platform (ServiceProvider#1).
The second stage is shown in Fig.3.6.

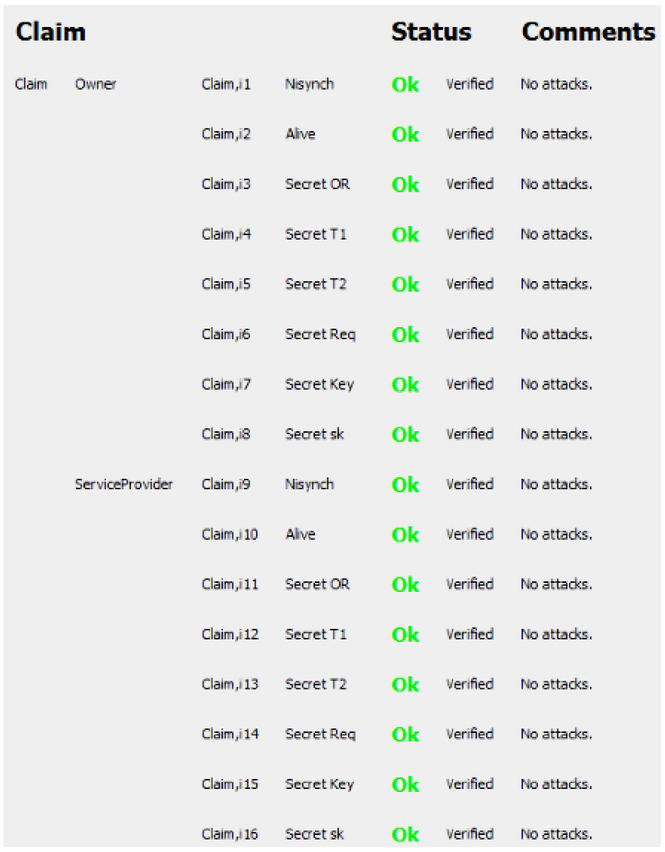
The final stage is shown in Fig. 3.7, where Owner#1 receives the message sent by ServiceProvider#1 and checks the integrity of its content through the match functions. The three generated traces show that message synchronization exists between the interacting parties. Moreover, it also proves the aliveness and reachability of parties.

IV. ANALYSIS OF RESULTS

This section presents a performance analysis of BBSMAP. The following assumptions are made for the analysis. We will see the mapping between the Scyther test results and the fundamental security requirements: Mutual authentication, Confidentiality, Integrity, Authorization, Accountability and nonrepudiation

A. MUTUAL AUTHENTICATION

The claimi1, claimi2, claimi9, and claimi10 indicated in Algorithm 1 on lines 21, 22, 42, and 43, respectively, are used to verify that the platforms of the owner and the service provider are in sync. For communications to be sent and received by their intended recipients in the same sequence, they must be synchronized. In addition, the use of public key infrastructure (PKI) and the protection of private keys fulfill the criteria for identifying platforms, and as a result, mutual authentication occurs between the two parties in this scenario.



Claim				Status		Comments
Claim	Owner	Claim,i1	Nisynch	Ok	Verified	No attacks.
		Claim,i2	Alive	Ok	Verified	No attacks.
		Claim,i3	Secret OR	Ok	Verified	No attacks.
		Claim,i4	Secret T1	Ok	Verified	No attacks.
		Claim,i5	Secret T2	Ok	Verified	No attacks.
		Claim,i6	Secret Req	Ok	Verified	No attacks.
		Claim,i7	Secret Key	Ok	Verified	No attacks.
		Claim,i8	Secret sk	Ok	Verified	No attacks.
	ServiceProvider	Claim,i9	Nisynch	Ok	Verified	No attacks.
		Claim,i10	Alive	Ok	Verified	No attacks.
		Claim,i11	Secret OR	Ok	Verified	No attacks.
		Claim,i12	Secret T1	Ok	Verified	No attacks.
		Claim,i13	Secret T2	Ok	Verified	No attacks.
		Claim,i14	Secret Req	Ok	Verified	No attacks.
		Claim,i15	Secret Key	Ok	Verified	No attacks.
		Claim,i16	Secret sk	Ok	Verified	No attacks.

Fig. 3.8: Output of Scyther claims test

B. NONDISCLOSURE OF INFORMATION

Secrecy is ensured by claiming claimi3 to claimi8 and claimi11 to claimi16 in lines 23–28 and 44–49, respectively, for the timestamps, the request, the output of the request, and the private request key, and the session key transferred. Throughout the exchange, the information is kept private.

C. INTEGRITY

The integrity of the request, time stamps, and output of the request are checked using the hash function and match operations. Change in the data is detected by the match routines provided in lines 18–20 and 39.

D. AUTHORIZATION

It is only authorized parties that have access to the decryption keys can access the information that has been communicated throughout the interaction. This is shown by the fact that claimi7 in line 27 and claimi15 in line 48 both ensure the confidentiality of the transferred session key. Furthermore, mutual authentication ensures that contact between trustworthy individuals, protecting the system from illegal access.

E. NONREPUDIATION

It is possible to identify the author of communication transmitted over a platform using its signature. Platforms sign messages with their private keys in BBSMAP, which allows for the potential of action tracing.

The two requirements are also addressed in the BBSMAP framework. Through many simultaneous broadcasts, BBSMAP accounts for agent loss in a system. Furthermore, as previously described in this section, its soundness is shown by a formal verification demonstration using Scyther. BBSMAP, on the other hand, is the only protocol among the efforts under consideration that confirm correctness by formal verification.

V. PERFORMANCE EVALUATION OF BBSMAP

When evaluating the performance of BBSMAP, it is necessary to estimate the total number of operations performed and compare it to the performance of the other protocols. An assessment approach generally acknowledged by the scientific community is utilized to compare BBSMAP and the other protocols. As stated by (Lei et al. 2008), the computing cost of an asymmetric operation (A) is equal to one point operation, which is comparable to 1000 symmetrical operations (S) and 10,000 hash operations (H), respectively (H). As a result, every asymmetrical, symmetrical, and hash operation is assessed as one point operation, one 0.001 point operation, and one 0.0001 point operation, respectively.

The following systems, in addition to BBSMAP, are estimated to have a similar total number of operations: (Garrigues, C et al. 2010, Garrigues Olivella, C et al. 2008, Hacini, S et al. 2008, Hedin, Y et al. 2015). These schemes were selected. They are comparable to BBSMAP because they are similarly based on cryptographic methods, and hence they were considered for inclusion. Depending on N, which is the number of platforms visited by the MA to gather results in MAS, the total number of operations is determined. Table shows the total number of operations for each scheme, which is determined separately.

Scheme	Total operations for system with N platforms
Guan et al., 2007 [31]	$(11A + 4H) \times N + 6A + 2H = 11.0004 \times N + 6.002$
Srivastava and Nandi, 2014 [34]	$(6A + 2S + 6H) \times N + 4A + S + 2H = 6.0205 \times N + 4.0012$
Ouardani et al., 2007 [33]	$12A \times N = 12 \times N$
Venkatesan et al., 2010 (XRC) [27]	$(3A + H) \times N + (1A + H) = 3.0001 \times N + 1.0001$
Geetha and Jayakumar, 2011 [32]	$(8A + 3H) \times N + (2A + H) = 8.0003 \times N + 2.0001$
BROSMAP	$(6A + 4S + 6H) \times N = 6.0046 \times N$

The growth in the number of operations as the number of service providers N grows is investigated to assess the performance of protocols. Fig. 5.2 depicts the change in the total number of approximated operations as the number of approximated operations grows.

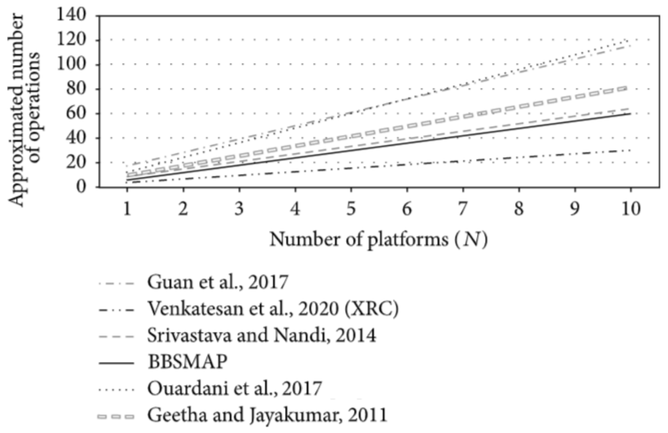


Fig. 5.2: The performance analysis of the proposed method BBSMAP in compression with the other existing protocols

The asymmetrical operations performed are the most important factor affecting performance since they incur the greatest computational cost. The protocol in (Hacini, S et al. 2008) is based only on asymmetrical operations, but the protocol in (Garrigues, C. et al. 2010) is based on both asymmetrical and hash operations (Garrigues, C. et al. 2010). As a result, they exhibit the highest rise in operations as the number of activities grows.

Garrigues Olivella and colleagues (2008) developed a technique that employs asymmetrical and hashed operations to guarantee anonymity, secrecy, integrity, and nonrepudiation protection. Furthermore, the protocol's performance is not only reliant on N, but it is also dependent on the number of trustworthy hosts that the MA encounters along its voyage. Assuming that just one trust host is visited to estimate the number of operations for this scheme, we can calculate the least number of operations performed by the system using this approximation. Fig. 5.2 illustrates that, compared to the other schemes, the hybrid technique proves to be more efficient. It also has a lower total computing cost than the other systems. While both (Hedin, Y., and Moradian, E. 2015) use multi hop approaches, BBSMAP utilizes broadcast architecture, and as the number of platforms rises, BBSMAP outperforms (Hedin, Y., and Moradian, E. 2015) in terms of performance.

VI. IMPLEMENTATION OF PROPOSED SYSTEM

The implementation of an example of an LBS application that uses BBSMAP for securing its communication on the well-known JADE (Bedi, P et al. 2008, Beheshti, S et al. 2007) platform is provided to verify the viability of the BBSMAP protocol and its implementation on the JADE platform is provided.

JADE (Java Agent Development Framework) is a software system that is entirely designed and written in Java. The purpose of the structure is to make the use of multi-agent systems easier by employing FIPA-compliant middleware and a collection of implicit devices such as the sniffer agent and the platform management console that aids in the investigation and organization of agent-based frameworks.

A. IMPLEMENTATION OF PROPOSED SYSTEM

Under the scenario described in Section 4, the program presents users with names of potential hotels, restaurants, and airline agencies close to the designated area. Users are presented with several alternatives and some information about each option, such as the average price and distance from their current location, to choose from. Four containers were developed to emulate the program, namely the Main Container, Containers 1 and 2, Containers 3 and 4, and the Containers 4 and 5; see Fig. 6. One container represents the owner platform, while the others represent the service providers' platforms, as seen in Fig. 6.5. Agents are employed on a mobile and a fixed basis. MAs known as owner agents are in charge of transporting the owner's request to the service providers and collecting the results. In addition to the owner MAs, other agents live in the service provider platform, interacting with them to process requests and exchange results. Owner1, Owner2, and Owner3 are three MAs of the type owner produced at the Main Container level: Owner1, Owner2, and Owner3.

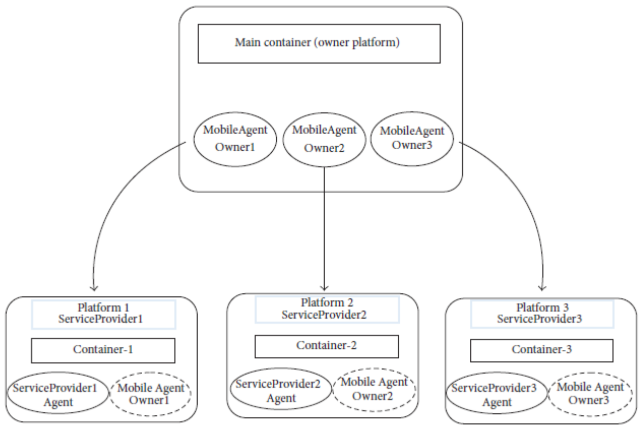
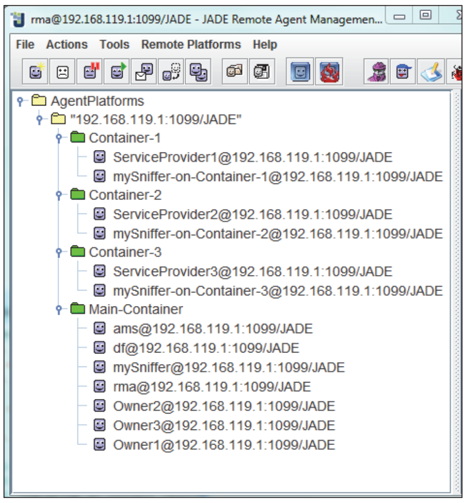


Fig. 6.5: Basic architecture of JADE setup


rma@192.168.119.1:1099/JADE - JADE Remote Agent Managemen...
File Actions Tools Remote Platforms Help

```
AgentPlatforms
  "192.168.119.1:1099/JADE"
    Container-1
      ServiceProvider1@192.168.119.1:1099/JADE
      mySniffer-on-Container-1@192.168.119.1:1099/JADE
    Container-2
      ServiceProvider2@192.168.119.1:1099/JADE
      mySniffer-on-Container-2@192.168.119.1:1099/JADE
    Container-3
      ServiceProvider3@192.168.119.1:1099/JADE
      mySniffer-on-Container-3@192.168.119.1:1099/JADE
    Main-Container
      ams@192.168.119.1:1099/JADE
      df@192.168.119.1:1099/JADE
      mySniffer@192.168.119.1:1099/JADE
      rma@192.168.119.1:1099/JADE
      Owner2@192.168.119.1:1099/JADE
      Owner3@192.168.119.1:1099/JADE
      Owner1@192.168.119.1:1099/JADE
```

Fig. 6.6: Simulated container agents and in JADE.

When a user requests a service, the request is broadcast, and each copy of the owner agent forwards the request to a separate service provider. At each service provider, three stationary agents (ServiceProvider1, ServiceProvider2, and ServiceProvider3) are created: ServiceProvider1, ServiceProvider2, and ServiceProvider3. Fig. 6.5 depicts the containers and agents in JADE before starting the interaction procedure.

When two parties communicate, JADE gives the user the option to sniff the messages passed between them. To track interactions between the owner agents and the service providers' agents, a sniffer agent, called my Sniffer, is installed at each container. Fig. 6.6 depicts sniffed messages exchanged between the owner and service providers throughout processing a user request and transmitting the resulting results to the user.

Fig. 6.7 depicts the content of one of the detected messages from Owner1 to ServiceProvider1, which was sent to ServiceProvider1. The content of the message is encrypted, and it can only be read when it has been properly decrypted at the intended destination.

The outcome of successful interaction between the owner and service providers is shown in Fig. 6.7. The final findings gathered by MAs are presented to the user and include the hotel's name, its distance from the location, and an estimate of the average price for a one-night stay based on historical data.

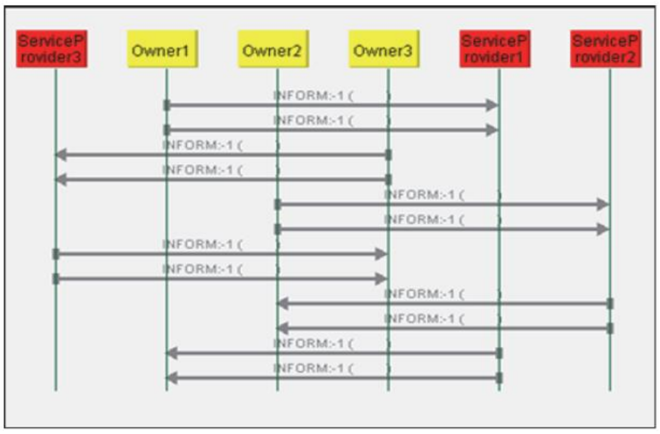
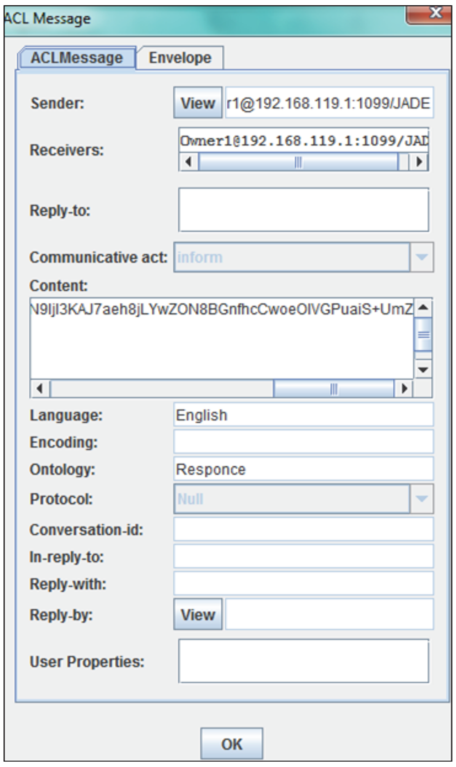


Fig. 6.7: Traced exchanged messages in JADE

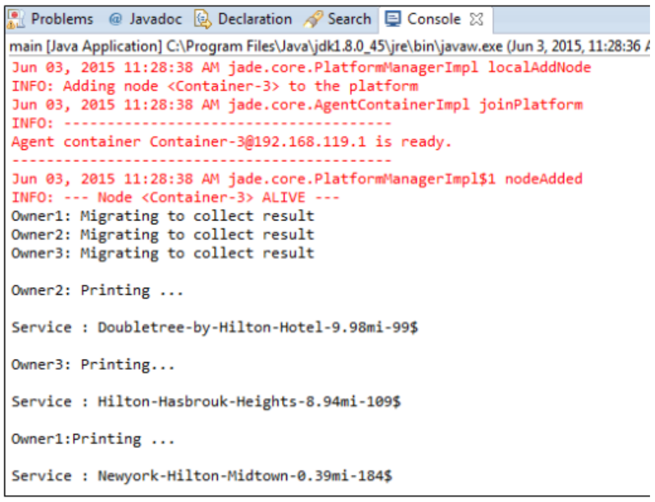
VII. CONCLUSION AND FUTURE RESEARCH

The authors introduced Broadcast-based Secure Mobile Agent Protocol (BBSMAP) for real-time distributed applications in this paper. Asymmetric and symmetric cryptography is used in conjunction with the proposed broadcast architecture of MA to deliver efficient, high-performance, and secure distributed computing applications. Mobile agent systems will benefit from the proposed protocol, which introduces the broadcast architecture, in which several copies of the agent are distributed to various service providers at the same time. As such, it fulfills the essential security needs of mutual authentication and authorization, integrity; confidentiality; accountability and nonrepudiation; and secrecy, accountability, and nonrepudiation. BBSMAP has also been shown to be resistant to a wide range of attacks, including replay, man-in-the-middle, masquerade, unauthorized access, and modification assaults, among others.



Sender:	r1@192.168.119.1:1099/JADE
Receivers:	Owner1@192.168.119.1:1099/JAD
Reply-to:	
Communicative act:	inform
Content:	N9IjI3KAJ7aeh8jLYwZON8BGnfhcCwoeOIVGPuaiS+UmZ
Language:	English
Encoding:	
Ontology:	Responce
Protocol:	Null
Conversation-id:	
In-reply-to:	
Reply-with:	
Reply-by:	
User Properties:	

Fig. 6.8: Sniffed message from the service provider ServiceProvider#1 to the Owner Owner#1



```
main [Java Application] C:\Program Files\Java\jdk1.8.0_45\jre\bin\javaw.exe (Jun 3, 2015, 11:28:36 A
Jun 03, 2015 11:28:38 AM jade.core.PlatformManagerImpl localAddNode
INFO: Adding node <Container-3> to the platform
Jun 03, 2015 11:28:38 AM jade.core.AgentContainerImpl joinPlatform
INFO: --------------------------------------
Agent container Container-3@192.168.119.1 is ready.
--------------------------------------------
Jun 03, 2015 11:28:38 AM jade.core.PlatformManagerImpl$1 nodeAdded
INFO: --- Node <Container-3> ALIVE ---
Owner1: Migrating to collect result
Owner2: Migrating to collect result
Owner3: Migrating to collect result

Owner2: Printing ...

Service : Doubletree-by-Hilton-Hotel-9.98mi-99$

Owner3: Printing...

Service : Hilton-Hasbrouk-Heights-8.94mi-109$

Owner1:Printing ...

Service : Newyork-Hilton-Midtown-0.39mi-184$
```

Fig. 6.9: Final results displayed to the user

VIII. CONCLUSION AND FUTURE RESEARCH

The authors introduced Broadcast-based Secure Mobile Agent Protocol (BBSMAP) for real-time distributed applications in this paper. Asymmetric and symmetric cryptography is used in conjunction with the proposed broadcast architecture of MA to deliver efficient, high-performance, and secure distributed computing applications. Mobile agent systems will benefit from the proposed protocol, which introduces the broadcast architecture, in which several copies of the agent are distributed to various service providers at the same time. As such, it fulfills the essential security needs of mutual authentication and authorization, integrity; confidentiality; accountability and nonrepudiation; and secrecy, accountability, and nonrepudiation. BBSMAP has also been shown to be resistant to a wide range of attacks, including replay, man-in-the-middle, masquerade, unauthorized access, and modification assaults, among others. Furthermore, the broadcast design allows the system to continue operating fault-tolerant even if one of the agents is deliberately murdered or otherwise rendered inoperative.

With the help of the Scyther verification tool, a formal verification test was carried out, which demonstrated that the protocol was proper in terms of meeting the various security standards and providing protection against the various harmful assaults. Another step in the process was conducting performance analysis to see how efficient the suggested protocol was compared to other relevant protocols. The results of the performance investigation revealed that the amount of operations performed by BBSMAP is appropriate in comparison to the security measures it offers. Finally, JADE is used to construct a scenario involving an LBS application that utilizes BBSMAP to secure its connection with other LBS applications. Implementing the proposed protocol using JADE demonstrates the practicality of the protocol and its capacity to offer protection for users.

IX. REFERENCES

[1]. Adane, D &Sathe, S 2009, “A security model for data storing and data collecting agents,” Int. J. Comput. Sci. Netw. Security. (IJCSNS), vol. 9, no. 4, pp. 265-271.

[2]. Aderounmu, GA, Oyatokun, B & Adigun, M 2006, “Remote Method Invocation and Mobile Agent: A Comparative Analysis,” Issues in Informing Science & Information Technology, vol. 3.

[3]. “Aglets Software Development Kit” 2012, Available from: http://aglets.sourceforge.net/.
[4]. Ahn, G-J, Mohan, B & Hong, S-P 2007, “Towards secure information sharing using role-based delegation,” Journal of network and computer applications, vol. 30, no. 1, pp. 42-59.
[5]. Akyazi, U & Uyar, ASE 2008, “Distributed intrusion detection using Mobile Agents against DDoS attacks,” Proceedings of the 23rd International Symposium on Computer and Information SciencesISCIS”08, pp. 1-6.
[6]. Amazon Elastic Compute Cloud 2013, Available from: https://aws.amazon.com/ec2/.
[7]. Amazon Relational Database Service 2013, Available from: https://aws.amazon.com/ rds/.
[8]. Amazon Route 53 2013, Available from: <https://aws.amazon.com/route53/>.
[9]. Ametller, J, Robles, S & Ortega-Ruiz, JA 2004, “Self-Protected Mobile Agents,” Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems Volume 1, pp. 362-367.
[10]. Bagga, P, Hans, R & Sharma, V 2017, “A Biological Immune System (BIS) inspired Mobile Agent Platform (MAP) security architecture,” Expert Systems with Applications, vol. 72, pp. 269-282.
[11]. Bamasak, O & Zhang, N 2004, “A secure proxy signature protocol for agent-based M-commerce applications,” Proceedings of the Ninth International Symposium on Computers and Communications, vol. 1, pp. 399-406.
[12]. Bamasak, O & Zhang, N 2006, “Reputation management and signature delegation: A distributed approach,” Electronic Commerce Research, vol. 6, no. 3, pp. 227-263.
[13]. Bamasak, O, Zhang, N & Edwards, D 2005, “Di Signcryption: an integration of agent-based signature delegation with distributed reputation management scheme,” Proceedings of the 19th IEEE International Symposium on Parallel and Distributed Processing, pp.
[14]. Batarfi, O 2011, “Protecting mobile agents against malicious hosts using dynamic programming homomorphic encryption,” International Journal of Science & Emerging Technologies, vol. 1, no. 1, pp. 1-6.
[15]. Bedi, P & Gaur, V 2008, “Trust-Based Prioritization of Quality Attributes,” International Arab Journal of Information Technology (IAJIT), vol. 5, no. 3, pp. 223 – 229.
[16]. Beheshti, S, Ghiasabadi, M, Sharifnejad, M & Movaghar, A, 2007, “Fault-tolerant mobile agent systems by using witness agents and probes,” in Proceedings of the Fourth International Conference on Sciences of Electronic, Technologies of Information and Telecommunications. pp. 1 5.
[17]. Benachenhou, L & Pierre, S 2006, “Protection of a mobile agent with a reference clone,” Computer Communications, vol. 29, no. 2, pp. 268278.
[18]. Bergamo, P, D’Arco, P, De Santis, A & Kocarev, L 2005, “Security of public-key cryptosystems based on Chebyshev polynomials,” IEEE Transactions on Circuits and Systems I: Regular Thesis, vol. 52, no. 7, pp. 1382-1393.
[19]. Burkle, A, Essendorfer, B, Hertel, A, Muller, W & Wieser, M 2006, “A test suite for the evaluation of mobile agent platform security,” Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology, pp. 752-756.
[20]. CentOS 2013, Available from: https://www.centos.org/.

[21]. Chuanrong, Z & Yuqing, Z 2009, “Secure mobile agent protocol by using encryption schemes,” Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, pp. 108-112.

[22]. Chung, Y-F, Chen, T-S & Lai, M-W 2009, “Efficient migration access control for mobile agents,” Computer Standards & Interfaces, vol. 31, no. 6, pp. 1061-1068.

[23]. Fatima, FB, Tounsi, W, Karoui, K & Ghazala, HB 2009, “A distributed multi-level anomalies detection system using the mobile agent approach,” Proceedings of the Global Information Infrastructure Symposium, pp.1-4.

[24]. Garrigues, C, Robles, S &Borrell, J 2008, “Securing dynamic itineraries for mobile agent applications,” Journal of Network and Computer Applications, vol. 31, no. 4, pp. 487-508.

[25]. Garrigues, C, Robles, S, Borrell, J & Navarro-Arribas, G 2010, “Promoting the development of secure mobile agent applications,” Journal of Systems and Software, vol. 83, pp. 959-971.

[26]. Garrigues Olivella, C & Robles, S 2008, “Contributions to mobile agent protection from malicious hosts,”Bellaterra: Universitat Autònoma de Barcelona.

[27]. Hacini, S, Cheribi, H & Boufaïda, Z 2008, “Dynamic adaptability using reflexivity for mobile agent protection,” International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol. 2, no. 11, pp – 3860 3865.

[28]. Hedin, Y & Moradian, E 2015, “Security in Multi-Agent Systems,” Procedia Computer Science, vol. 60, pp. 1604-1612.

[29]. Hijazi, A & Nasser, N 2005, “Using mobile agents for intrusion detection in wireless ad hoc networks,” Proceedings of the Second IFIP International Conference on Wireless and Optical Communications Networks, pp. 362-366.

[30]. Hong, X 2009, “Efficient threshold proxy signature protocol for mobile agents,” Information Sciences, vol. 179, no. 24, pp. 4243-4248.

[31]. Jiang, Y, Xia, Z, Zhong, Y & Zhang, S 2004, “Defend mobile agent against malicious hosts in migration itineraries,” Microprocessors and Microsystems, vol. 28, no. 10, pp. 531-546.

[32]. Jianxiao, L & Lijuan, L 2009, “Research of Distributed Intrusion Detection System Model Based on the mobile agent,” Proceedings of the International Forum on Information Technology and Applications, vol. 2, pp. 53-57.

[33]. Kocarev, L &Lian, S 2011, “Chaos-based Cryptography,” Theory, Algorithms and Applications, vol. 354, Springer.

[34]. Kumari, VV, Kumar, YA, Raju, K, Ramana, K & Prasad, R 2007, “Policy-based controlled migration of mobile agents to untrusted hosts,” Proceedings of the Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing, vol. 1, pp. 550-553.

[35]. Kun, G & Sumei, J 2010, “Research on the application of mobile agent in intrusion detection technology,” Proceedings of the 2nd International Conference on Computer Engineering and Technology, vol. 6, pp. V6-549-V546-553.

[36]. Lei, S, Liu, J, Deng, C & Xiao, J 2008, “A novel free-roaming mobile agent security protocol against colluded truncation attacks,” Proceedings of the International Conference on Information and Automation, pp. 348-353.

[37]. Lei, S, Liu, J & Xiao, J 2008, "A novel free-roaming mobile agent security mechanism by trusted computing technology," Proceedings of the International Conference on Computer Science and Software Engineering, vol. 3, pp. 721-724.

[38]. Li, X & Chen, K 2004, "Identity-based proxy-sign cryption scheme from pairings," Proceedings of the IEEE International Conference on Services Computing, pp. 494-497.

[39]. Linna, F & Jun, L 2010, "A free-roaming mobile agent security protocol against colluded truncation attack," Proceedings of the 2nd International Conference on Education Technology and Computer, vol. 5, pp. V5-261-V265-265.

[40]. Liu, L, Logan, KP, Cartes, DA &Srivastava, SK 2007, "Fault detection, diagnostics, and prognostics: software agent solutions," IEEE Transactions on Vehicular Technology, vol. 56, no. 4, pp. 1613-1622.

[41]. Lu, T & Fu, M 2006, "Using mobile agents for object sharing in P2P networks", Proceedings of the First International Conference on Innovative Computing, Information and Control, vol. 1, pp. 741-744.

[42]. Manvi, SS & Venkataram, P 2007, "Mobile agent-based approach for QoS routing," IET Communications, vol. 1, no. 3, pp. 430-439.

[43]. Marikkannu, P &Jovin, 2011, "A secure mobile agent system against tailgating attacks," Journal of Computer Science, vol. 7, no. 4, p. 488.

[44]. Marikkannu, P, Jovin, JA & Purusothaman, T 2011a, "An enhanced mobile agent security protocol," European Journal of Scientific Research, vol. 51, no. 3, pp. 321-331.

[45]. Marikkannu, P, Jovin, JA & Purusothaman, T 2011b, "Fault-tolerant adaptive mobile agent system using dynamic role-based access control," International Journal of Computer Applications, vol. 20, no. 2, pp. 1-6.

[46]. Marikkannu, P, Jovin, JA & Purusothaman, T 2012, "Self-Protected mobile agent Approach for Distributed Intrusion Detection System against DDoS Attacks," International Journal of Information and Electronics Engineering, vol. 2, no. 4, p. 606.

[47]. McDonald, JT, Yasinsac, A & Thompson III, WC 2004, "Mobile agent data integrity using multi-agent architecture," Air Force Inst Of Tech Wright-Pattersonafb Oh.

[48]. Micciancio, D 2010, "The first glimpse of cryptography's Holy Grail," Communications of the ACM, vol. 53, p96.

[49]. Mo, X-l, Wang, C-d & Wang, H-b 2009, "A Distributed intrusion detection system based on mobile agents," Proceedings of the 2nd International Conference on Biomedical Engineering and Informatics, pp. 1-5.

[50]. Ou, C-M, Wang, Y-T & Ou, C 2008, "Dynamic role assignment based on X. 509 PMI mechanism for mobile agent systems", Proceedings of the International Conference on Machine Learning and Cybernetics, vol. 7, pp. 3699-3703.

[51]. Peters, J 2005, "Integration of mobile agents and web services," Proceedings of the 1st European Young Researchers Workshop on Service-Oriented Computing, vol. 5, pp. 53-58.

[52]. Qing, X 2010, "The structure design of a new distributed intrusion detection system," Proceedings of the 2nd International Conference on Computer Engineering and Technology, vol. 7, pp. V7-100-V107-103.

[53]. Shao, M-H & Zhou, J 2006, "Protecting mobile-agent data collection against blocking attacks," Computer Standards & Interfaces, vol. 28, no. 5, pp. 600-611.

[54]. Shen, Z & Wu, X 2008, "A trusted computing technology-enabled mobile agent system," Proceedings of the International Conference on Computer Science and Software Engineering, vol. 3, pp. 567-570.

[55]. Songsiri, S 2005, "A new approach for computation result protection in the mobile agent paradigm," Proceedings of the 10th IEEE Symposium on Computers and Communications, pp. 575-581.

[56]. Srivastava, S & Nandi, GC 2014, "Self-reliant mobile code: a new direction of agent security," Journal of Network and Computer Applications, vol. 37, pp. 62-75.

[57]. Su, CJ 2008, "Mobile multi-agent based, distributed information platform (MADIP) for wide-area e-health monitoring," Computers in Industry, vol. 59, pp. 55-68.

[58]. Venkatesan, S &Chellappan, C 2008, "Protection of mobile agent platform through attack identification scanner (AIS) by malicious identification police (MIP)," Proceedings of the First International Conference on Emerging Trends in Engineering and Technology, pp.1228-1231.

[59]. Venkatesan, S, Chellappan, C, Vengattaraman, T, Dhavachelvan, P & Vaish, A, 2010, "Advanced mobile agent security models for code integrity and malicious availability check," Journal of Network and Computer Applications, vol. 33, no. 6, pp. 661-671.

[60]. Wang, C, Han, Y & Li, F 2009, "A secure mobile agent protocol for m-commerce using self-certified proxy sign cryption," Proceedings of the Second International Symposium on Information Science and Eng, pp. 376-380.

[61]. Wang, M & Liu, Z 2005, "Identity-based threshold proxy sign cryption scheme," Proceedings of the Fifth International Conference on Computer and Information Technology, pp. 695-699.

[62]. Wang, S, Wang, J, Dang, C & Liu, A, 2007, "A trust evaluation method of mobile agent system," Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing, pp. 6317-6320.

[63]. Wu, X, Shen, Z & Zhang, H 2006, "The mobile agent security enhanced by trusted computing technology," Proceedings of the

[64]. International Conference on Wireless Comm., Networking and Mobile Computing, pp. 1-4.

[65]. Xian, H-q & Feng, D-g 2007, "Protecting mobile agents' data using trusted computing technology," Journal of Communication and Computer, vol. 4, no. 3, pp. 44-51.