2009 Third International Conference on Network and System Security

A Multi-agent Security Architecture

Rossilawati Sulaiman, Dharmendra Sharma, Wanli Ma, Dat Tran
Faculty of Information Science and Engineering,
University of Canberra,
Australia
{Rossillawati.Sulaiman, Dharmendra.Sharma, Wanli.Ma,Dat.Tran}@canberra.edu.au

***Abstract*—This paper presents a multi-agent security architecture, which utilizes the agent characteristics to cater for security processes in online communications. The Multilayer Communication approach (MLC) [1] is used to determine the security processes, which uses cryptography protocols to secure data and communication channel. Agents are skilled to perform certain tasks. At the Sender's host, agents interact with each other to secure a message to be sent to the Recipient, including encryption, digital signature, and hash code. A mobile agent is used to carry the encrypted messages as well as the agent's code to the Recipient's host. Our approach also provides mechanisms to verify the authenticity, confidentiality and the integrity of the code and data that arrived at the Recipient's host. The message and the code are authenticated, the code is executed to perform tasks to recover the plaintext.**

Keywords-e-health; security, cryptography; multi-agent; mobile agent;

I. INTRODUCTION

Ever since the first time people started to become aware of the value of information, they are also conscious about the underlying security issues. Reliance to the Internet as a medium of communications to exchange and share information has become so prevalent. However, the Internet network is exposed to many security threats. An organization needs to protect their system so that their valuable information would not be harmed or stolen by irresponsible intruders. A number of security technologies exist as a mean to secure the communication environment. However, there exist many threats to the Internet. For example network attacks, information breaches, and malicious software, which is a program that is purposely created to perform illegal operation on the computer system, such as viruses and worms. These threats can cause severe damages to the computer systems as well as the information. The information might be stolen or modified and may cause undesirable consequences. Therefore, it is important to secure online communications which uses the Internet.

Using these problems as a motivation, we proposed a security model that caters for the security needs for online communications between two nodes (say n1 and n2). This model can be adapted for different kinds of domains. In our model, focus is given to strengthen the application and transport layers on the Open System Interconnection (OSI) model [2] (Figure 1).

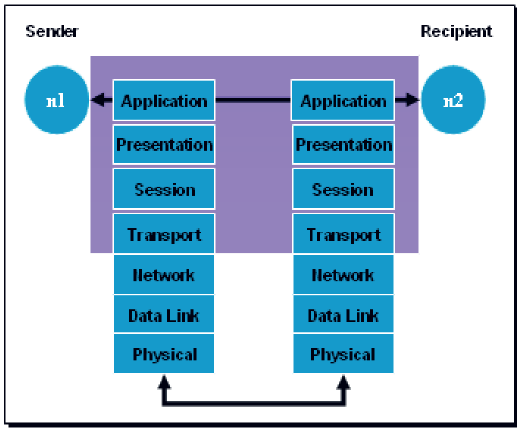

Figure 1. Securing application and session layers in the OSI Model.

In our approach, cryptography protocols are used and applied on data at the application layer, as well as on the communication channel using SSL/TLS at the transport layer. The Multilayer Communication approach (MLC) [1] is used to determine the security processes that will be applied to the messages exchanged between nodes. The multi-agent system approach is chosen to cater for the security processes at the Sender and Recipient's hosts. The rest of the paper is organized like the following: Section 2 discuses the MLC approach. Section 3 discusses the related work. Next, the proposed security model is discussed in Section 4. Then, Section 5 describes the implementation, followed by the result in Section 6. Finally this paper is concluded with a summary in Section 6.

II. THE MLC APPROACH

The MLC approach is discussed in this section to ease the understanding of our proposed security model.

Sensitive data and the sensitivity: During a communication, say between nodes n1 and n2, different types of information are exchanged. There is information that can be considered as sensitive, or less sensitive, or no sensitive at all. Sensitive information are those that should not be revealed to public [3]. Whether the information is sensitive or not is based on the importance or the values of the information. It is important to assess the value of the information and the consequence it will cause if it is made public or fall to the wrong hand. The levels of sensitivities of the data refers to the degrees of loss or risks that might happen if the data are disclosed to a party that does not have any authority, or which has a lower level access to the data.

TABLE I. THE CLASSIFICATION AND SECURITY MECHANISMS IN THE MLC APPROACH

Layer of Communication	**E.g. of Types of Data**	**Security Mechanisms**	
Layer 1: ***Extremely Sensitive data*** Doctor⇔Doctor Doctor⇔Patient Doctor⇔Nurse Nurse⇔Patient	Patient's personal information, medical history, diagnosis, test result, current treatment and prescriptions	Data and channel security	192-bit key and longer
Layer 2: ***Highly sensitive data*** Paramedic⇔SC	Patient's personal information, medical information: allergic, blood pressure, current condition	Data security (using wireless network)	80-bit key to 191-bit key
Layer 3: ***Medium sensitive data*** Doctor⇔SW Nurse⇔SW	Patient's personal information, medication information	Channel security or data security	112-bit key to128-bit key
Layer 4: ***Low sensitive data*** SA ⇔ all users	Information on the application system, user account, non-medical related information such as IT technical problem	Channel security or data security	80-bit to less than 112-bit of key
Layer 5: ***No sensitive data or Public*** The public	*Open channel:* general information on the organization, health, diseases, FAQ, public reports, and services available	Secure open channel: ID and password	-
	Secure open channel: any user that wants to get access or contact information to any sensitive information (e.g.: a researcher)		

A name, a place, and a meeting time, are probably less sensitive than a name, an address, and types of diseases, which can be associated together to make someone assumes that that person with that name is actually having that types of illness. Other example that fall into this category is information in electronic business operation. For instance, a supplier's bid must not be revealed to other suppliers, or trade secret information must not be disclosed to other parties or their competitors. There is also information that can be considered as less sensitive. For example an employee communicates with a system administrator of a company regarding his technical problem on his computer, or a nurse communicates with a social worker regarding a patient's name and his ward's number. The less sensitive information, if it is made public, will not bring much risk to the company, but still it must not be revealed to public.

No sensitive information includes general information of a company or general information on health or diseases that can be made public.

Communication classification: We have proposed a Multilayer Communication approach (MLC) [1] that characterizes the communication between different users into five different layers: Layer 1 to Layer 5, namely Extremely Sensitive, Highly Sensitive, Medium Sensitive, Low Sensitive and No Sensitive Data, in descending order. We determined the level of sensitivity of the data based on the roles or users who are exchanging the data. Using an e-health domain as an example, we identified the users in the e-health as Doctor, Patient, Nurse, Social Worker (SW), Paramedic, System Coordinator (SC), and System Administrator (SA), such as described in Table1. Different security mechanisms can be provided at each layer depending on the sensitivity of the data. Highest security mechanisms can be applied to the extremely sensitive information, while low security mechanisms can be applied to the low sensitive information. MLC treats every communication differently based on the sensitivity of the information being exchanged unlike Virtual Private Network (VPN) or Secure Shell (SSH). By providing different security mechanisms in every layer, an organization can choose its own security mechanisms flexibly, depending on cost and performance.

Security mechanisms: In each layer, a user can choose either to apply data security, channel security or both. Data security involves encryption, digital signature and hash of data, while channel encryption provides a secure SSL/TLS channel to transfer the data. A range of key lengths for data encryption is provided for every layer.

The key length: Each layer is associated with the length of the symmetric keys that are used for encryption (described in Table1). Higher layer uses longer key length and lower layer uses shorter key length. As we can see in Table I, Layer 2 provides key lengths, ranging from 80-bit key to 191-bit key, which started with a rather short key length (from 80-bit). This is because we are taking into account communications with wireless devices that have low processing powers. Only data security is applied in Layer 2. The details of the MLC approach can be found in [13]. The next section describes the related work for this research.

III. RELATED WORK

The use of multi-agents system has become a well accepted paradigm to support online communication to exchange messages over the network. The agents are used to cater for the communication processes as well as the security mechanisms applied to the message before transmitting the message to the intended recipient. Cryptography protocols such as encryption/decryption, digital signature, and hash message are chosen as the security mechanisms. For example, Yenta [4] is a multi agent application for matchmaking purposes that introduces users with similar interests. Yenta uses symmetric encryptions to secure data and PKI to exchange keys. Data is also verified using hash message. The X-Security prototype [5] is another example of multi agent application that provides security for online communication. It uses central server that issues certificates for every agent as a proof of identity as well as for session key exchanged.

Moreno et al. [6] use multi-agent system to provide medical services. The system authenticates every agent that wants to enter the system. Communications between agents through container or host are through SSL-enabled channel. To communicate with the main container, every agent must deal with a broker agent, who checks other agent's identity through a signature, through a public key mechanism to prevent other agents from stealing or forging their identities. Alsinet el al. [7] developed a multi-agent system architecture to monitor medical protocols. Agents interact to provide different services. Messages are encrypted and signed. 3DES algorithm is used for encryption and exchanged using public key RSA mechanism. Certificates are used to distribute users' public keys.

These works stated above are without doubt providing security to the communications and data transmission and hence protecting the privacy of the data based on the cryptographic protocols. However, these approaches view all communications between users to be the same, and hence, all communications will be secured by the same security processes. In our approach, through MLC, we provide a range of security mechanisms through multiple layers of communication. The different combinations of security mechanisms in MLC provide flexibilities to the organization to choose the most suitable security processes in terms of cost and performance. For example, we can choose SSL, which is cheaper, rather than data encryption [8]. Moreover, low processing power devices, such as PDAs and smart phones may require only data encryption with a range of appropriate key lengths, which can save processing resources. Next, we discuss the proposed security model and the security mechanisms.

IV. THE PROPOSED SECURITY MODEL

A. *The Multi-agent Approach*

Our proposed model is based on the multi-agent system approach. An agent is defined as a computer system that can receive input and output from the environment, autonomous and flexible [9]. The agent-based technology is chosen in this research because of the advantages of being able to support the implementation of complex programs in the distributed environment, such as the Internet. An agent can represent a user and do tasks on behalf of the user. Agents can be given knowledge to do specific task and interact with each other in order to solve a given task. For instance, in a Sender and Recipient communication, an interface agent interacts with Sender and takes messages from the interactions. The agent then forwards the message to another agent to apply appropriate security mechanisms on it. To send the message, a mobile agent is instantiated and dispatched to carry the message to the Recipient's host.

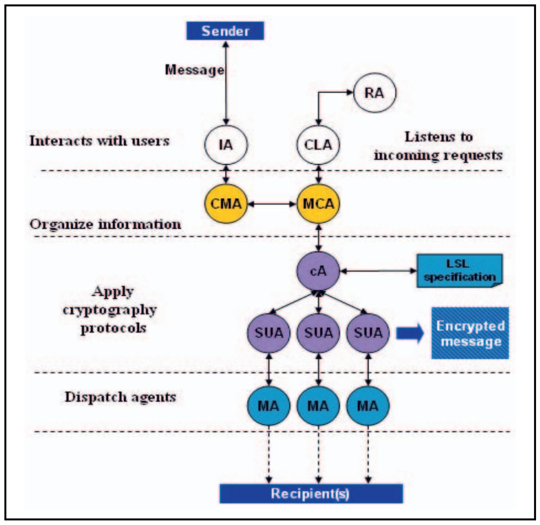

Figure 2. The proposed security model.

Before sending the message, Sender agent first makes a request to send a message to the Recipient's agent. If there is no answer, Sender agent may take necessary steps to keep the message and try to send it again later, or delete it if it is already exceeded a certain period of time with a notice to the user. This shows that the agent is autonomous, and it does not need user's intervention and can do tasks on its own. In addition, agent systems are extendible. A new agent can be created instantly and added to the existing system to perform a certain task, and terminate itself after finishing its tasks without reconfiguring the system.

The mobile agent is used because of its ability to perform tasks on behalf of the owner at remote hosts. A mobile agent from Sender can be dispatched to carry sensitive information and its code to the Recipient's platform. There, the agent negotiates with an agent that resides in the Recipient's host to execute its code. In addition, mobile agents are robust, in a sense that if the Recipient's platform is shut down while the agent is still there, the agent may take necessary actions to migrate back or terminate its activities [10]. It can send a notice to the home platform about its situation and terminate if required.

Figure 2 describes the proposed multi-agent based security model in a layered structure. In our model, every agent is skilled to perform certain tasks. The agents coordinate with each other to perform security processes in order to secure data or channel. There are eight main types of agents described as Interface Agent (IA), Communication Manager Agent (CMA), Multilayer Communication Agent (MCA), Communication Listener Agent (CLA), Receiver Agent (RA), crypto Agent (cA), SetUp Agent (SUA) and MobileAgent (MA).

IA provides users with user interfaces and interacts with them. IA first gets the user ID and password to authenticate the user into the system. If the user is authorized, IA provides an interface to create new message and view any received message. To create a new message, the user may write the text message and specify the recipient(s). IA then takes the list of recipient(s) and the text message, and sends it to CMA that manages the information. CMA and MCA interact with each other to organize the message to be sent to the recipient(s).

When the user (in this case, Sender) composes a new message, MCA makes a request to the Recipient's host to send a message. If the Recipient's host agrees, then MCA notifies cA, which in turn applies appropriate security mechanisms to the message. As mentioned before, every message must be secured according to MLC approach. MLC specifications for each layer are stored at the Layer Specification Library (LSL). LSL contains available algorithms, key lengths, encryption mode and padding information for symmetric key encryption. For each recipient, cA creates an instance of SUA that will be responsible to apply the cryptography protocols according to the security specification. After finish applying the security mechanisms, SUA creates an instance of MA that will be used to carry the encrypted message to the Recipient's host. There, MA executes the code to decrypt the message.

At the Recipient's host, CLA listens to any request from other users. When a request is received, and the Sender is authorized (the certificate of the sender is in the list of trusted user) then, CLA creates an instance of Receiver Agent (RA) that will wait for the message from the Sender. RA will be responsible to entertain the mobile agents arrive at the host.

B. Control of the data

In our approach, we focus on how Sender can securely transfer data to Recipient while maintaining control over the data. The 'control over the data' can be described as (1) if the message carried by Sender Agent is seized by an attacker, the attacker still cannot recover the plaintext, (2) recipient or any other third party does not need to know the details of the decryption processes to recover the plaintext. One way for Sender to gain control over the data, is to keep part of the requirements for the decryption processes secret, such as part of the agent's code, or parameters used for decryption. The parameters for decrypting the ciphertext are kept with Sender until he/she knows that MA needs it. The parameters contain the symmetric key that encrypts the plaintext, hash of plaintext (for Recipient to verify the plaintext), and the information about the key, such as the types of the algorithms, key length, and encryption mode.

A token, which is an encrypted random number, is carried by MA to the Recipient's host. It is used as a 'phone home' [11] mechanism for the agent to tell Sender that it wants the information kept at the Sender's side to continue its task. The 'phone home' method is a way for the mobile agent to contact Sender from the Recipient's host, to tell Sender that it needs necessary information for the decryption processes.

The authenticity of the agent's code is verified once it arrives at the host. If the agent's code is valid, it is executed. At this point, the agent is ready to perform the decryption process to recover the plaintext. This is where the agent sends the token back to Sender.

C. The Communication Protocols

Consider that we have two parties communicating between Patient and Doctor. Patient wants to send a plaintext (P) to Doctor. We assume that certificates exchange has been done through a secure channel that makes Patient trusted by Doctor. Patient is represented by SetUp Agent (SUA), which dispatches a mobile agent (MA) to the Doctor's side. At the Doctor's side, Doctor is represented by Receiver Agent (RA), which is responsible to interact with the incoming MA. The communication protocols between SUA, MA and RA are described in Figure 3.

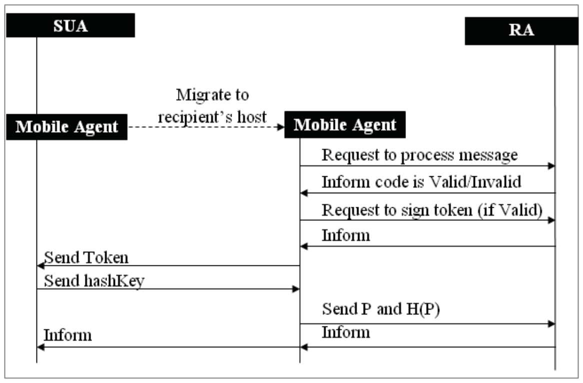
SUA
RA
Migrate to recipient's host
Mobile Agent
Mobile Agent
Request to process message
Inform code is Valid/Invalid
Request to sign token (if Valid)
Inform
Send Token
Send hashKey
Send P and H(P)
Inform
Inform

Figure 3. Communication protocols for SUA, MA, and RA

Consider that MCA has accepted an "Agree" answer from CLA. MCA tells cA about a new message to send, and cA creates an instance of SUA. After preparing the message, SUA creates and dispatches an instance of MA to carry the data to the Doctor's host. Once arrived, MA initiates a communication with RA. MA requests RA to process the carried data. RA then processes the data and informs MA whether the code is valid or not to be executed. If the code is valid, MA retrieves the token, and requests RA to sign it. Once signed, MA sends the token back to SUA. SUA processes the token, and if the token is not tampered, SUA sends information for decryption process to MA. When the information is received, MA performs the decryption process to recover P. When P is recovered, MA sends P and H(P) to RA for verification. RA informs MA for the verification result and finally, MA informs SUA and terminates.

D. The Proposed Security Mechanism

This section discusses the proposed security mechanism, mainly the cryptography protocols that are used to secure data and mobile agent's code. We refer to the previous example in Section 4(C). The following symbols will be used throughout this paper:

- Recipient/Doctor's public/private keys: *(pubKr, privKr)*
- Sender/Patient's public/private keys: *(pubKs, privKs)*

- Symmetric keys: *K1, K2*;
- Disposable public and secret key: *(Kp, Ks)*
- Information about the key: *lsl*
- Plaintext, *P*; Hash of P, *H(P)*
- Agent's code: *Cd*; Hash of *Cd*, *H(Cd)*
- Ciphertext, *C*; Signature, *S*
- A Random number *Rand*; Token*, T*

1) *At the Sender's host, SUA does the following:*
   i. Generates two symmetric keys (*K1* and *K2*)
   ii. *K1* is used to encrypt *P* to produce a ciphertext, *C*. *K1* is also used to encrypt *Rand* to generate a token, *T* that will be carried by the agent.
   C = E(P)K1, T = E(Rand)K1
   K1 is kept at the Sender's side until SUA receives *T* from MA at the remote host.
   iii. Generates asymmetric keys, public and secret key (*Kp*, *Ks*). *Ks* is used to encrypt *hashKey* (detailed at point 2(v)) when SUA receives *T* from the MA. *Kp* will be kept in *Cd* and sent to the Patient's host. The generation of *(Kp, Ks)* is one time per communication session*. (Kp, Ks)* will be disposed once the communication session is over to avoid being used by any third party for the next communication session.
   iv. Create a .jar file containing the agent's code to be executed at the Recipient's host, called *Cd*. Sign *Cd* with *privKs* to produce a signature, *S*, which is used to verify that *Cd* is from Sender. *S* will be encrypted so that it cannot be removed and replaced by any party except RA.
   S = E(Cd)privKs
   v. Encrypts *Cd*, *S*, and *T* with *K2* to produce *Ciphercode*.
   Ciphercode = E(Cd, S, T)K2
   vi. To allow only RA to retrieve *K2*, it is encrypted with *pubKr* together with the hash of *Cd*, *H(Cd)* to produce *Cipherkey*.
   Cipherkey = E(K2, H(Cd))pubkr
   vii. Save *C*, *Ciphercode* and *Cipherkey* into a file. Establishes SSL connection if necessary, and dispatch a mobile agent to send the file to the Recipient's host.

2) *At the Sender's host, SUA does the following:*
   i. RA uses *privKr* to decrypt *Cipherkey*
   D(Cipherkey) privKr = K2, H(Cd)
   ii. Then, *K2* is used to decrypt *Ciphercode*
   D(Ciphercode)K2 = Cd, S, T
   iii. To check that *Cd* is indeed come from Sender, *S* is verified against *Cd*
   iv. The integrity of *Cd* is also checked using *H(Cd).*
   v. If *Cd* is not violated, then *Cd* is executed.
      a. Get *T* and request RA to sign it.
      b. Once signed, *T* is sent back to SUA, to indicate that MA is ready for the decryption process.
      c. Upon receiving *T*, SUA verifies the signature and decrypts *T* with *K1*. If *T* is not modified, then SUA encrypts *K1*, information on *K1 (lsl),* and *H(P)* with *Ks* to produce *hashKey*. The corresponding *Kp* is stored within the code *Cd*.
      hashKey = E(K1,lsl,H(P))Ks
      Then, *hashKey* is sent back to MA.
      d. *hashKey* is received from SUA
      e. Again at the Recipient's host, *Cd* retrieves *Kp*, and use it to decrypt *hashKey* to obtain *K1*, *lsl*, and *H(P).*
      f. *Cd* uses *K1* and *lsl* to decrypt the *ciphertext* to get plaintext, *P*.
      g. Afterwards, both *P* and *H(P)* will be forwarded to RA for verifying purposes.
   vi. Finally, RA verifies *P* by calculating a new *H(P)* from *P*, and compare it with the one forwarded earlier. If proved valid, keep *P*.

The next sections onwards describe the implementation of the security model.

V. IMPLEMENTATION

We have implemented the proposed security model as a proof-of-concept, using Jade [12], a Java-based FIPA-compliant agent platform. Each user's machine runs the Jade agent platform. Each platform is able to accept foreign agents (which are the mobile agents) by using inter-platform mobility service. Each agent is created with behaviours that represent the agent tasks. Classes of IA, CLA, CMA, MCA, and cA are instantiated by every platform, so that each platform can act as both Sender and Recipient. For agent interactions, we implement it using FIPA-compliant ACL Language and ontology.

When a message is received from MCA, cA looks up at the LSL for a suitable algorithm and key length. cA creates an instance of SUA that is responsible for applying appropriate security mechanisms on the plaintext. Figure 4 describe pseudocodes for SUA.

```
Class SUA:(address,RAname,lsl,textfile)
Start
Create .jar file (Cd)
Initialize privKs, pubKr
privKs=ExtractPrivK(Sender's keystore)
pubKr=ExtractPubK(recipient's certificate).
Generate (K1, K2) and (Ks, Kp)
Initialize Ciphertext, Token
Ciphertext =Encrypt(textfile, K1)
Generate a random number, rand
Token=Encrypt(rand,K1)
Take the agent's code Cd:
    Initialize S, H(p), H(Cd)
    S = Sign(Cd,privKs)
    H(p)= computeHash(textfile)
    H(Cd)=computeHash(Cd)
    Initialize Ciphercode, Cipherkey
```

```
   Ciphercode=Encrypt((Cd, S, T)K2
   Cipherkey=Encrypt(H(Cd), K2)pubKr
   Save Ciphertext, Ciphercode, and
   Cipherkey into a file
CreateNewMobileAgent(address, RAname)
DispatchMA(address, message file)
IF mobile agent request for hashKey
   Initialize hashKey
   hashKey=Encrypt(HP),K1, LSL)
   Send hashKey to mobile agent
END-IF
End
```

Figure 4. The pseudocode for SUA.

SUA is responsible for applying suitable security mechanisms according to the LSL value. It first makes a .jar file called Cd, to package the agent classes that are to be executed in the recipient's platform. Then, it takes the recipient's certificates and extracts the public key. Then it generates appropriate symmetric keys according to the LSL specification.

MA carries the message to the recipient's host, and there it will communicate with RA. Figure 5 describes the pseudocode for MA. When Cd is proved to be valid by RA, it is executed. Cd extracts information from hashKey to obtain the hash of the plain text, H(P), K1 for decryption, and LSL. Then K1 is loaded and recreated based on the LSL specification. Afterwards, using K1, Cd decrypts the ciphertext to obtain the actual text message such as described in Figure 6.

```
class MA:ACLMessage msg)
START
Send REQUEST ("Process-Message")
Receives INFORM for result ("Process-
Message").
IF result == Valid (indicating that both the
agent's code and the signature are valid)
   Send REQUEST ("Sign-Token")
   Send signed Token to SUA
   Receive hashKey from SUA
   Un-jarred Cd.
   Send REQUEST to the AMS to execute Cd
   (parameter: hashKey, ciphertext)
END-IF
IF ("Finish-Encrypting") is received
   Send INFORM to SUA to return result from
   Cd
END-IF
Terminate itself
END
```

Figure 5. The pseudocode for MA.

```
START
Initialize K1, plaintext
K1= recreateKey(K1,LSL)
plaintext = Decrypt(Ciphertext,K1)
Send REQUEST to RA ("Check-plaintext")
   (parameter: plaintext, H(P))
Receives INFORM with result (indicating
plaintext is valid or not)
Send INFORM to MA ("Finish-Encrypting") to
return result
Terminate itself
END
```

Figure 6. The pseudocode for Cd.

RA will be instantiated when a REQUEST to send a message is received. The RA's name will be returned to the requester along with an AGREE or REJECT message. RA will be in charged of communicating with MA in the process to decrypt a message. Figure 7 presents the pseudo code for RA.

```
Class RA:(ACLMessage msg)
START
IF ("Process-Message") is received
  Split(message)=Ciphertext,      Ciphercode,
  Cipherkey
  Initialize privKr, pubKs
  privKr=ExtractPrivK(Rec's keystore)
  pubKs=ExtractPubK(Sender's cert).
  Decrypt(Cipherkey, privKr)=K2,H(Cd)
  Decrypt(Ciphercode,K2)= T, S, Cd.
  Validate(S,Cd)pubKs
  Initialize newH(Cd)
  newH(Cd)=computeHash(Cd)
  Compare(newH(Cd),H(Cd)).
  IF S and H(Cd) == valid
    send INFORM to MA containing  "valid"
  END-IF
END-IF
IF ("Sign-Token") is received
   Sign(Token,privKr)
END-IF
IF ("Check-plaintext") is received
  Initialize newH(P)
  newH(P)=computeHash(P)
  IF Compare(newH(P),H(P))==true
     Put plaintext in a messageQueue
     Send notice to user
   END-IF
END-IF
END
```

Figure 7. The pseudocode for RA

VI. RESULT

A. Experimental Setup

For the experiment, we create a controlled environment by using 2 PCs connected to each other in a LAN. Each computer is equipped with Pentium IV, 3 GHz CPU, and 1 GB RAM. We calculate the execution times (in millisecond) for:

(1) Agent-based communications
(2) Non-agent based communications using Java socket

In (1), we measure six types of executions using JADE-agent with plaintexts sizes range from 50Kb to 300Kb. Since the mobile agent that carries the data to the Recipient's host is not returning to the Sender's host, the Round Trip Time (RTT) time cannot be measured. Therefore, we measure the time starting from the *preparation of the data* (generating Ciphertext, Cipherkey, and Ciphercode), sending the data across by MA, until Sender receives the token back from MA to ask for hashKey. We used the following five MLC security settings and one with no security:

i. Layer 1: data security (AES-256) and channel security
ii. Layer 2: data security (Blowfish-184)
iii. Layer 3: data security (AES-128)
iv. Layer 4: data security (Blowfish-96)
v. Communications that use SSL channel security only (option for Layer 3 and Layer 4) using mobile agent to transfer the plaintext. We use JADE-HTTPS setting such as in [14].
vi. Communications with no security setting to transfer plaintexts, to measure the overhead cost of the security processes

In (2), we use Java-socket with plaintexts sizes range from 1Mb to 10Mb. The same security setting is used as in (1) and six execution times are measured started from the preparation of data (same as in (1)), sends it through sockets to the Recipient, and until Recipient sends a simple message back to Sender to ask for hashKey. We also calculate Java-socket communication with no security. For all data security, we choose CBC mode, and PKCS7 padding type.

B. Simulation Results

1) Agent based Communication

Figure 8 shows the execution times taken for agent-based communication.

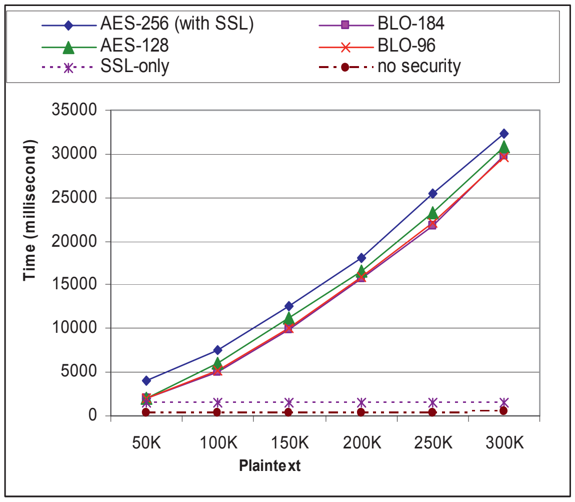

Figure 8. The execution time for agent-based communciations

The result shows that the communication that uses only channel security performs better than the other channel and/or data security agents. This is because, in the channel security communication, only SSL channel is establish, and it does not require any process for encryption, decryption, hash, or sign any plaintext. Moreover, the size of the plaintext transferred by the mobile agent does not change. For the channel and/or data security communications, the data security processes add up to the total data size that should be carried by MA, which is almost twice the size of the original plaintext. For example, for a plaintext with the size of 250Kb, the total size of data (containing ciphertext, cipherkey, and ciphercode) carried by the mobile agent to the recipient's platform is about 464 Kb.

From the result, we can also see that Blowfish algorithm has the superiority against AES-256 and AES-128, regardless of the key sizes. There is no effect of changing the Blowfish's key lengths as both Blowfish-184 and Blowfish-96 give almost similar throughputs for all plaintext sizes. The same result also found in [15].

2) Java socket MLC Communication

Figure 9 shows the execution times for Java-socket communications. The same pattern of the previous results can be observed. The result shows that the communications with the SSL-channel security perform better than the other communications. This is because there is no need to encrypt/decrypt, sign, or hash the plaintext. We can also find that Blowfish algorithm performs faster than both AESs.

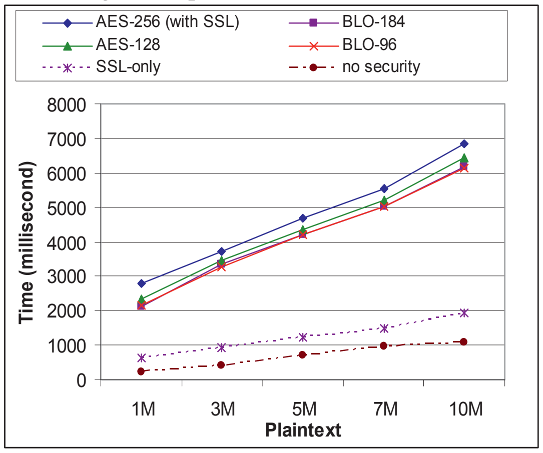

Figure 9. The execution time for Java-socket communications

For the overhead calculations, we refer to Table II and Table III.

TABLE II. OVERHEAD OF 300KB COMMUNICATIONS

Security setting	Percentage
SSL only	283.4
Layer 4	7468.2
Layer 3	7750.6
Layer 2	7504.7
Layer 1	8152.6

In Table II, we compare the result of the *no security* communications with 300Kb plaintext, with the other security settings for the agent-based communications. We found out that the SSL-only communication has over 200% overhead than the no security communication, Layer 4 and Layer 2 have the average of 7500% overhead, Layer 3 has 7750% increment and Layer 1 has the highest overhead with over 8000%.

For Java-socket, we compare the result of the *no security* communications with 10Mb plaintext, with the other security settings (depicted in Table III). We found out that the SSL-only communication has lower overhead with only 78%, Layer 4 and Layer 2 have the average of 468% overhead, Layer 3 has 492% and Layer 1 has the highest overhead with 530%.

Overheads projected by the agent-based communications are higher than the Java-socket communications. This is expected because it takes longer time for the multiple layers of communications among agents (refer to Figure 2), using ACL Language to complete the security protocols. Overhead of 10Mb communications

Security Setting	Percentage
SSL only	78.4
Layer 4	466.0
Layer 3	492.1
Layer 2	470.2
Layer 1	530.3

From the discussion, we can conclude that Layer 1 has the highest overhead compared to the other layer, because of the use of both data and channel security mechanisms. For Layer 2, 3, and 4 that use data security, the selection of the algorithms is also an important issue. Because Blowfish is a faster algorithm than AES [16] regardless of the key lengths, we found out that Layer 2 and 4 have performed better than Layer 3 that used AES-128 algorithm. Therefore, the selection of the algorithm types should be taken into account when setting up the MLC. However, because MLC is a flexible approach, any algorithm can be chosen, as long it can give better performance to the communications.

VII. CONCLUSION

A security model base on multi-agent system is presented in this paper. The agents are skilled with knowledge to cater for the security processes to secure online communications. The mobile agent is used as a supporting tool to carry sensitive data. Cryptographic protocols are used to secure data as well as the mobile agent code. Using this security model, Sender can gain control over the plaintext, because Recipient or any other third party does not know the details of the decryption processes. Experiments have been conducted and tested using Jade platform as a proof-of-concept.

Results showed that agents incur higher cost compared to the traditional method. However, it gives a much better control on security to the initiator of the communication with assuring security of the channel and at the Recipient's node.

The result also showed that Layer 1 communication has the highest overhead, due to data and channel security applied in the layer. The SSL-only communication has the lowest overhead. For Layer 2, 3, and 4 that use data security, we suggest that the selection of the algorithms characteristics should be taken into account before choosing them to fit into MLC.

The layered structure improves efficiency based on the level of security decides at different levels. There can be a significant gain of efficiency for encryption, decryption, and transmission with a careful selection based on needs of the appropriate layers in the security model. This security model may fit in any types of domain. As our ongoing work, we are implementing the agent on mobile devices.

ACKNOWLEDGMENT

Rossilawati Sulaiman is supported by a join scholarship from the Public Service Department (PSD) of Malaysia and the Ministry of Higher Education of Malaysia.

REFERENCES

[1] Sulaiman, R., Sharma, D., Ma, W., & Tran, D. (2008). A Security Architecture for e-Health Services. The 10th International Conference on Advanced Communication Technology (ICACT), Phoenis Park, Korea.

[2] B.N. Jain and A. K. Agrawala. Open System Interconnection: Its Architecture and Protocols. McGraw-Hill, 1993.

[3] Pfleeger, C. P., & Pfleeger, S. L. (2006). Security in computing (4th edition): Prentice Hall PTR.

[4] L. N. Foner (1996). "A Security Architecture for Multi-Agent Matchmaking," In: IC/lAS96.

[5] Petr Novak and Milan Rollo and Jiri Hodik and Tomas Vlcek: Communication Security in Multi-agent Systems. In Multi-Agent Systems and Applications III. Springer, 2003. ISBN 3-540-40450-3.

[6] A. Moreno, D. Sánchez, and D. Isern, "Security Measures in a Medical Multi-Agent System," Artificial Intelligence Research and Development (Proc. CCIA 03), vol. 100 of Frontiers in Artificial Intelligence and Applications, I. Aguiló, L. Valverde, and M.T. Escrig, eds., IOS Press, 2003, pp. 244–255.

[7] Alsinet, T., R. Bejar, et al. (2000). A Multi-agent system architecture for monitoring medical protocols. Proc of the fourth international conference on Autonomous agents. Barcelona, Spain, ACM.

[8] French, B. (2006). Implementing soa security. 2007, from ftp://ftp.software.ibm.com/software/tivoli/whitepapers/Implementing SOAsecurityG507-1918-00.pdf.pdf

[9] Jennings, N.R., Sycara, K., Wooldridge, M.: A roadmap of agent research and development. Autonomous Agents and Multi-Agent Systems 1(1), 7–38 (1998)

[10] Danny, B. L., & Mitsuru, O. (1999). Seven good reasons for mobile agents (Vol. 42, pp. 88-89): ACM.

[11] Michael, J. G., & Brian, D. M. (1999). Protecting the integrity of agents: an exploration into letting agents loose in an unpredictable world (Vol. 5, pp. 10-17): ACM.

[12] Jade, Java Agent Development Framework. http://jade.cselt.it, 2006.

[13] Sulaiman, R., Huang, X., & Sharma, D. (2009). E-health Services with Secure Mobile Agent. Paper presented at the Seventh Annual Communication Networks and Services Research Conference, Los Alamitos, CA, USA.

[14] Exposito, J.A., Ametller , J., & Robles, S. (2003). Configuring the JADE HTTP MTP. http://jade.cselt.it/doc/tutorials/HTTP_UAB.html

[15] Mousa, A. (2005).Data Encryption Performance Based on Blowfish, 47th International Symposium ELMAR-2005 focused on Multimedia Systems and Applications, pp. 131- 134, Zadar, Croatia

[16] D. S. Abdul. Elminaam, H. M. Abdul Kader and M. M. Hadhoud (2009), Performance Evaluation of Symmetric Encryption Algorithms,Volume 8, number 8, Communications of the IBIMA, 2009, pages 58-64, http://www.ibima.org/pub/journals/CIBIMA/volume8/v8n8.pdf