Journal of Computer Science 8 (5): 637-647, 2012
ISSN 1549-3636
© 2012 Science Publications

Enhancing Security in E-Health Communications using Multi-Agent System

[1]Rossilawati Sulaiman,
[2]Dharmendra Sharma, [2]Wanli Ma and [2]Dat Tran
[1]School of Computer Science,
Faculty of Information Science and Technology,
National University of Malaysia, 43600, Bangi, Selangor, Malaysia
[2]Faculty of Information Science and Engineering,
University of Canberra, Bruce, 2601, ACT, Australia

Abstract: Problem statement: In this study, we address the security needs in online communications, specifically in the e-health domain. We focus on how to provide different security strengths to different types of communications in e-health, where each communication transmits different types of information with different levels of sensitivity. **Approach:** The Multi-Agent System (MAS) approach is used to develop an agent-based system that can cater for distributed processes. We use the agents' characteristics such as *autonomous*, *interactive*, *extendible* and *mobile* to handle the security processes for users in different environments and devices. We integrate different types of encryption algorithms with different security strengths in order to provide different security needs. **Results:** We present our security model called MAgSeM that consists of eight agents, which are skilled to complete its goal as well as the overall system goals autonomously. **Conclusion:** We conclude that MAgSeM security model is suitable not only for the e-health domain, but also other domains that practices online communications.

Key words: Multi-agent system, security, cryptography protocols, e-health

INTRODUCTION

Nowadays, the emergence of computer applications that support online communications has a remarkable impact on how businesses are carried out. For example, e-services, which are services that are delivered via the Internet, to provide online services to consumers (Douligeris and Serpanos, 2007). Applications of e-services can be seen in e-health, Internet banking, retailing and electronic auctions. However, these applications also bring about security issues and the needs for reliable and robust security technologies to cater for security threats. These threats take advantage of computer systems vulnerability to cause damage to the systems.

In e-health, services are delivered online through the Internet such as via e-mail, web applications and videoconferencing (Liu *et al*., 2008). This clearly shows that sensitive information is delivered online, which motivates users to protect their privacy when doing online transactions (online communication will hereafter be referred to as *online communications*). Current security technologies such as Secure Socket Layer/Transport Layer Security (SSL/TLS) (Dierks and Allen, 1999), Secure Shell (SSH, 2003), IPSecurity (RFCs 2401-2411 and RFC 2451 for IPSec), or Virtual Private Network or VPN (Pardoe and Snyder, 2005) are undoubtedly robust and secure to be used as a security measure to secure online communications. Table 1 summarizes these technologies, which are based on the Transmission Control Protocol and the Internet Protocol (TCP/IP) model.

In general, each technology offers a list of available encryption algorithms that are used to encrypt messages in transit during the communication sessions. However, these technologies do not cater for *different types of communication needs* in an organization. For example, consider that a company uses SSL for its secure communications. If the company needs to change the security *strength* of the SSL channel (the security strengths indicate the key lengths of the ciphers) to be stronger or weaker, it cannot be flexibly provided to the organization. The person in charge, for instance the Security Administrator, needs to reconfigure the systems to change the security setting.

Corresponding author: Rossilawati Sulaiman, School of Computer Science, Faculty of Information Science and Technology, National University Malaysia, 43600, Bangi, Selangor, Malaysia

Table 1: Summary of security technologies

Security technology	Mechanisms provided
SSL/TLS	Protect the transport layer Provide authentication, digital signature and encryption to the message in transit using a collection of algorithms in the negotiated cipher suite.
SSH	Protect the transport layer Provide authentication, data integrity and digital signature. Data encryption is provided during SSH session, that exchange a symmetric key.
IPSec	Protect the Internet layer Provide authentication, data integrity, authentication of origin and anti replay protection. Data confidentiality is provided using symmetric key algorithms.
VPN	VPNs provides various tunnelling protocols such as authentication, data encryption, data integrity and digital signature

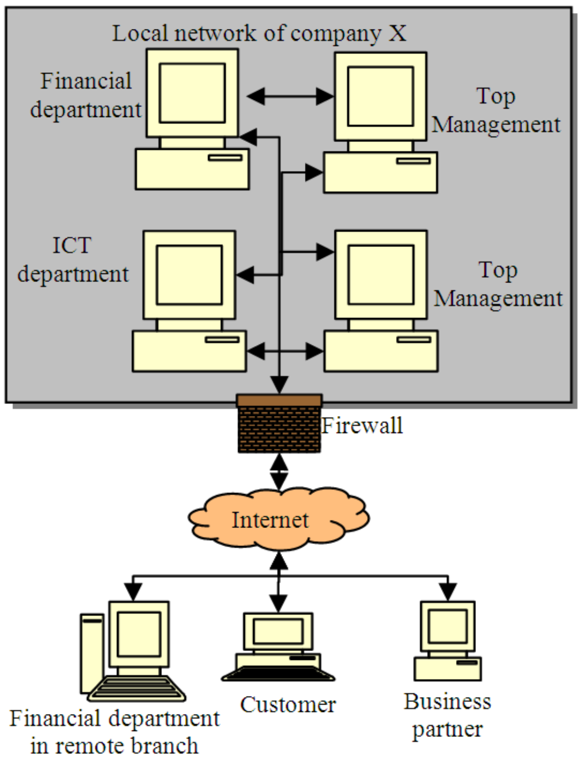

Fig. 1: Example of a network in company X

Communication scenario: Suppose that company X needs different security levels of protections to secure different kinds of communications. Consider that the company has distributed employees, business partners and customers (as described in Fig. 1). Different users that are involved with the company have different goals and purposes when communicating with other users. There is a need within the company, that for each communication, *different security mechanisms* are required so that only the most sensitive information can be secured with the highest level of security mechanisms, medium sensitive information can be secured using medium level of security mechanisms and low sensitive information can be secured using low level of security mechanisms.

For example, a Top Management might want to send two different messages to two different users (1) a message containing a secret about business strategic plan to another user in the Top Management level; (2) a technical problem in his/her personal computer to a user in the ICT Department. If considering the company's need to secure the message according to the sensitivity of the message, these two different messages require two different sensitivity levels.

The first message can be considered as the most sensitive and the other is low sensitive. These two messages must be secured differently, the first with the highest level of security mechanisms and the second with low level of security mechanisms.

The need for stronger or weaker security strengths can be found in information classification standard, such as in ISO17799, 2007 that portrays Top Secret, Highly Confidential, Proprietary, Internal Use Only and Public Document. We can also see information classifications in EO12958 1995 SIGS, 2001, which classify information into Top Secret, Secret, Confidential and Restricted. For each and every classification, different level of security protection is needed for different types of information with different levels of sensitivity.

We are motivated to find the best way to secure different types of communications that transmit these different types of information, in such a way that it could provide different types of security strengths to the communications. If we investigate current security technologies, such as listed in Table 1, they cannot satisfy the requirement, because of the limitation they have on the configuration setting. They only allow all communication sessions to be secured with the same security strength. In other words, current technologies do not cater for the following requirements:

- Provide different security strengths to secure different types of communications
- Provide an automated and a flexible way for the system to cater for all kinds of users' needs without reconfiguring the system and
- Provide mechanisms to handle security for two communication users in different environments, such as PC to PDA communications.

METERIALS AND METHODS

Research materials:

Security mechanisms in MLC: We have proposed the Multilayer communication approach (or MLC) in

(Sulaiman *et al*., 2007). The purpose of the MLC model is to develop a flexible and secure communications system for domains like e-health. MLC classifies communication in e-health into five layers namely extremely sensitive, highly sensitive, medium sensitive, low sensitive and no sensitive. This classification is based on the different sensitivity of the information being exchanged during communications.

MLC provides three types of security mechanisms that are data security, channel security and both data and channel security. In data security, cryptography protocols such as encryption and decryption, hash message, as well as digital signature are used. In channel security, SSL is used to provide secure channel. For both data and channel security, cryptography protocols are imposed on the data and then transmitted over SSL.

Communication layers: For simplicity purposes, users that are involved in the communication in e-health are identified as Doctor, Patient, Nurse, Social Worker (SW), Paramedic, System Coordinator (SC) and System Administrator (SA).

Examples of communications includes a doctor at a hospital communicates with another doctor at another hospital; a patient or SW at home communicates with a doctor at the hospital; or a paramedic at a location of an accident communicates with SC at the hospital. The paramedic and SC work together, where information regarding a patient is sent by the paramedic using a PDA/smart phone and received by SC in the hospital for further action, such as preparing for a medical team while waiting for the patient to arrive at the hospital.

For a communication between two users say, Doctor and Nurse, the layer of communication or *com_layer* is identified. com_layer refers to the five layers of communications in the MLC model, which determines the security mechanisms and *symmetric key lengths* that will be applied to the information in a particular communication session. com_layer can be determined by using a default layer value (L_0), assigned as the following:

- Patient, Doctor and Nurse, L_0: Layer 1
- Paramedic and System Coordinator, L_0: Layer 2
- Social Worker, L_0: Layer 3
- System Administrator, L_0: Layer 4.

L_0 is assigned based on the sensitivity of the data each user may carry. Smaller L_0 is assigned to users that communicate extremely sensitive information, while a bigger value of L_0 is assigned to users that communicate low sensitive information. The rules to determine com_layer for a communication between a sender and a recipient are as follows:

- If sender's L_0 and recipient's are the *same,* then com_layer for that communication will be the recipient's L_0
- If sender's L_0 is *greater* than the recipient's, then com_layer for that communication is the sender's L_0
- If sender's L_0 is *smaller* than the recipient's, then com_layer for that communication is the recipient's L_0

Therefore, com_layer can be identified by comparing the sender's and the recipient's L_0s. The one with a larger value will be chosen as com_layer. For example, in Nurse (L_0 = 1) and SW (L_0 = 3) communication, the com_layer will be *Layer 3*. If both L_0s are the same (in Doctor and Patient communication, where L_0=1), then, the com_layer is equal to 1. After com_layer is identified, the security mechanisms for the communication can be determined, that is, whether the communication needs data security, or channel security, or both data and channel security. The com_layer is associated with the length of the symmetric key encryption algorithms (Sulaiman *et al*., 2011):

- Layer 1: key length =193-bit key and longer
- Layer 2: key length = 129-bit to 192-bit key (for wireless: 80-bit to 192-bit key)
- Layer 3: key length = 112-bit to128-bit key
- Layer 4: key length = 80 to less than 111-bit of key

We can see that the Layer 1's key lengths are longer than the other layers, so that the strongest security can be given to the communication.

MLC specification: The MLC Specification stores the security specifications, which describes the information about the symmetric key encryption. MLC specification is stored as a tuple containing four parameters:

<Algorithm, lengths, mode, padding>

Algorithm, lengths, mode and padding describe the types of algorithms for the symmetric key, the lengths of the key, encryption modes and encryption padding respectively.

Figure 2 gives an example of a set of MLC specifications. Examples of available algorithms that can currently be used are AES 256-bit and 192-bit representing Layer 1, Triple-DES 168-bit for Layer 2, AES 128-bit, Twofish 128-bit, TEA 128-bit and Blowfish 112-bit represents Layer 3 and Blowfish 80-bit represents Layer 4.

The encryption mode is cipher-block chaining mode (CBC) and padded with PKCS7 padding. The encryption algorithm is selected *randomly* after com_layer has been calculated as part of the communication process.

<AES-256-CBC-7>
<AES-192-CBC-7>
<3DES-168-CBC-7>
<AES-128-CBC-7>
<BLO-112-CBC-7>
<BLO-80-CBC-7>
<TWO-128-CBC-7>
<TEA-128-CBC-7>

Fig. 2: An example of MLC specifications

Any standard algorithms can be added to this specification. The method of selecting the algorithms presented here is more flexible compared to the method in the existing technologies such as presented in Table 1.

Multiagent-based approach: We are interested to use the multi-agent characteristics to cater for the security processes, so that MLC could be implemented as a multi-agent system. The characteristics of the agents include the following.

Agents can represent a user to handle security processes automatically. The security processes are done stage by stage and distributed because they needs resources from remote recipient such as permission to send a message as well as the private key of the recipient to perform cryptography protocols (this will be described later on).

The autonomy and interactive characteristics allows agents with different capabilities to secure the data, which involve interacting with the user/sender, organizing the information obtained from the user, listening to any connection from other users (recipients) and applying cryptography protocols. Agents have the ability to interact, coordinate and cooperate with each other in order to achieve the overall goal of the system (which is to send secure message to the recipients).

The extensible property of the agent allows it to be added or instantiated when a new communication is needed and deleted when the communication has ended. Thus, the agents can handle multiple communications at once.

The mobility characteristic of the agent can be used to carry the message across the network, which allows the agent to carry out tasks on behalf of the user/sender. From this discussion, we have found the required agent characteristics, namely autonomous, mobility, extendible and interactive.

Autonomous: The autonomy of an agent is a characteristic that allows agents to do the assigned tasks independently. Each agent has its own behaviour(s). The agent's actions or tasks assigned to the agent are performed from within the behaviours. Communications and coordination between agents, which are performed in order to complete a task, are done and handled through behaviours. When an agent is executed, it automatically performs its behaviour without being invoked by any external entity.

Agents are considered active. They have control on their actions as they have goals and rules. They know when to act, or update their states. Autonomous is performed by not providing the agent with call-backs function to its own object reference to other agents. Consequently, this will lessen any chance of other entities taking control of its services. Thus, control complexity is reduced and divided within the agents themselves (Bellifemine *et al*., 2007; Jennings, 2000).

Mobility: Mobility is the ability of the agent to migrate or move from home platform to another platform, carrying its code and data. Mobile agents can be characterised like the following (Braun and Rossak, 2005; Singh, 1998):

- Mobile agents are used in the wide-area and heterogeneous networks where there is no reliability on the connection or the security of the network
- The migration of the agents is initiated by the agent (or programmer)
- The agents migrate to access resources only available at the remote hosts

Extendible: Extendible focuses on adding or removing capabilities or skills of an agent to an existing system. Debenham (1999) defines extensibility as "*the abilities to easily add new functionality to a system, or upgrading any existing functionality*". To be extensible, MAS should be capable of performing new functionality that it is currently does not able to perform. A new agent representing a new system's functionality can be added to the system, without reconfiguring the whole system. In this research, we focus on the extensible functionalities, where the agent can be specialized or skilled and added to the existing system.

Interactive: The key element for multi-agent interaction and social organization is communication. Agents are able to cooperate and coordinate their actions to execute tasks through communications. Agent Communication Language (ACL) enables an agent to exchange data and information with other agents. FIPA-ACL and KQML (Bradshaw, 1997) are the most commonly used agent language.

KQML provides message formatting and message-handling standard to support knowledge sharing among agents in a run-time environment (Bradshaw, 1997). FIPA-ACL, which is based on the speech act theory, suggest that agent's actions are represented by messages (Bellifemine *et al*., 2007). Both ACLs have a distinct communication protocols known as *performative*, or communication act. This protocol uses speech act language such as *request*, *send*, *accept*, *rejec*.

RESULTS

Organizing the agents: We use *Organizational* Structure proposed by Nwana *et al*. (1996) as our coordination techniques for our multi-agent system. Organizational structure specifies a set of long-term responsibilities and interaction patterns for the agents (Durfee *et al*., 1987). The long-term responsibilities describe agent's functionality that guarantees long-term consistency and satisfactory result of the whole system performance.

While agents perform their responsibilities, there are agents that depend on other agent's partial solutions to perform their own responsibilities, which require to be informed of the other agent's partial solutions so that they can take further actions. An agent does not need any information that does not affect their actions. Therefore, apart from specifying the responsibilities, the organizational structure also decides particular agents that are interested or required a partial solution. When any exchanging process of partial solutions of one agent occurs, it will give effect to the other agent's actions (Luck, 2001).

The approach of organizational structure is to specify the agent's actions or functionalities and divide the problem search space among the agents, in such a way that particular agents are assigned to specific tasks. For further information on organizational structure, readers are referred to (Huhns and Gasser, 1989; Shoham and Tennenholtz, 1992; Wooldridge, 2002).

Identifying agents goals using organizational structure: We specify each agent's actions or functionalities that are assigned to them for completing an overall goal. According to Jennings (1996), in this technique, the actions of agents in solving goals can be expressed through a classical AND/OR graph (Mahanti and Bagchi, 1985).

We identify the overall multi-agent system goal and the sub goals using the AND/OR graph. These goals represent the agent's actions. Figure 3 and 4 show the AND/OR graph to illustrate the main goal (G and G') and their respective sub goals.

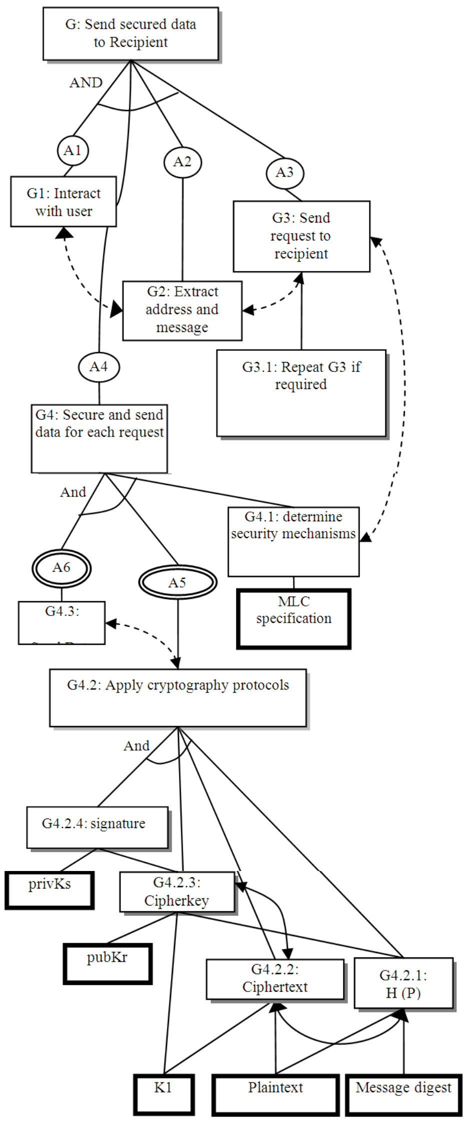
G: Send secured data to Recipient
AND
A1
A2
A3
G1: Interact with user
G3: Send request to recipient
G2: Extract address and message
A4
G3.1: Repeat G3 if required
G4: Secure and send data for each request
And
G4.1: determine security mechanisms
A6
A5
G4.3:
MLC specification
G4.2: Apply cryptography protocols
And
G4.2.4: signature
privKs
G4.2.3: Cipherkey
pubKr
G4.2.2: Ciphertext
G4.2.1: H (P)
K1
Plaintext
Message digest

Fig. 3: AND/OR graph for agent's goal G

The goal G is to *"Send Secured Data to Recipient"* while goal G' is to *"Listen to Incoming Requests"*. We divide G into G1 to G4 and assign an agent into each sub goal, labelled as A1 to A4. G' is divided into G'1 and we assign agent A7 to G'1. These sub goals may or may not contain another level of sub-goals. Agents that are drawn with double lines indicate agents that are instantiated to complete certain sub goals. For example, A5 and A6 are instantiated by A4 in order to complete G4.

An 'AND' node can only be completed, if all of the immediate sub-goals are completed. For example, G4.2 (*Apply Cryptography Protocols*) can only be completed if goals G4.2.1 to G4.2.4 are completed first. An 'OR' node can be completed by choosing either one of the sub-goals.

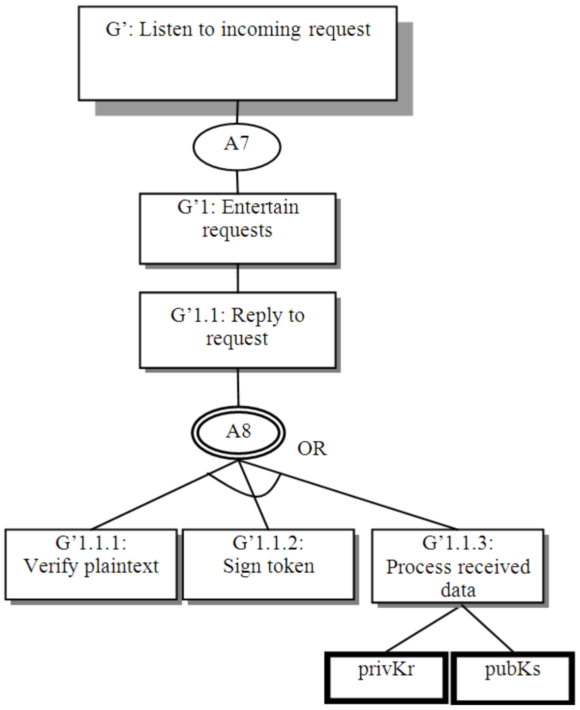
G': Listen to incoming request
A7
G'1: Entertain requests
G'1.1: Reply to request
A8
OR
G'1.1.1: Verify plaintext
G'1.1.2: Sign token
G'1.1.3: Process received data
privKr
pubKs

Fig. 4:AND/OR graph for the agent' goal G'

For example, A8 may entertain a request by performing either G'1.1.1 (*Verify plaintext*), or G'1.1.2 (*Sign token*), or G'1.1.3 (*Process received data*), provided that all conditions are met.

Both graphs illustrate the interdependencies between goals and data/resources, which are needed to solve the primitive goals. The solid arrows indicate interdependencies between goals and data/resources, which is drawn in bold lines.

The graphs also illustrate interdependencies between goals. The dotted arrows indicate interdependencies between G4.2 and G4.3, which are *Apply Cryptography Protocols* and *Send Data*. In an organizational structure, it is important for each agent to report each result of the sub-goals, or report that it has finished its goal to the other agents that need the result, so that other agents could use the results to determine their next actions. For example, A5 must report its end result to A6 and then only A6 can send the secured data to the recipient's side.

Organizing the Agents: The agents are renamed with meaningful names as follows:

A1: Interface Agent (IA): interacts and obtains data from the user authenticate a user to enter the system and acquire data from the user
A2: Data Organizer Agent (DOA): organizes the data received from the user such as the message and the recipient's address
A3: Multi-tasking Agent (MTA): makes a request to send a message to other users and keeps track of undelivered messages
A4: Crypto Agent (cA): provides all necessary information and parameters for the security processes
A5: SetUp Agent (SUA): applies cryptography protocols
A6: Mobile Agent (MA): carries secured data to the Recipient's platform
A7: Communication Listener Agent (CLA): listens to any incoming request
A8: Receiver Agent (RA): provides verification service to the agents that arrive at the platform

The agents cooperate with each other to perform security processes, by sending reports and messages (or partial results) to the other agents in order to achieve the overall goal. A report indicates that an agent has finished its task. The partial results are integrated in the process of generating an overall goal.

Proposed Multi-Agent based Security Model (MAgSeM): We proposed a multi-agent architecture called MAgSeM to cater for security processes for online communication in e-health, such as in Patient (as sender) to Doctor (as recipient) communication. The previous agents are integrated with another two agents, which are Server Agent (SvA): resides at the server's side to manage the authentication process and other requests from agents; and Decrypt Agent (DA): instantiated by MA to perform the decryption process at the recipient's side.

Fig. 5 shows the proposed multi-agent system called MAgSeM. The dotted lines show instantiated agents. At the sender's side, IA sends the ID, password and IP address of the sender to SvA to be authenticated (assumption is made that the certificates of all users have been exchanged beforehand. A security administrator in an organization, such as a hospital could be responsible for managing certification exchanges). SvA authenticates the user and if the sender is authorized, it sends the authentication result (valid/invalid) as well as a list of IP addresses of other users that have exchanged certificates with the sender.

Then, IA gets the data and address of the recipient(s) from the sender and gives it to DOA. DOA organizes the data into a file or plaintext and gives both plaintext and the address to MTA. The Secure Communication Layer is where the security processes are performed. At this layer, MTA sends a request to send a message to the intended recipient(s). cA prepares all necessary information for the security processes and determines the appropriate layer for the communication as described in MLC. SUA is instantiated by cA to apply the cryptography protocols. MA is instantiated by SUA to send the data to the recipient by migrating to the recipient's host.

At the recipient's side, CLA listens for any incoming request to send a message from MTA. CLA instantiates RA to entertain the incoming MA in case of a request. MA instantiates DA at the recipient's side to perform decryption processes. At the server side, SvA compares the ID and password with the one in the database. If they are matched, SvA stores the IP address of the user.

MAgSeM supports the implementation of MLC and provides a rather generic architecture that can be used in any type of domains. The agents use the autonomous, extensible, interactive and mobile characteristics and are able to coordinate and cooperate with each other to achieve the overall system goal.

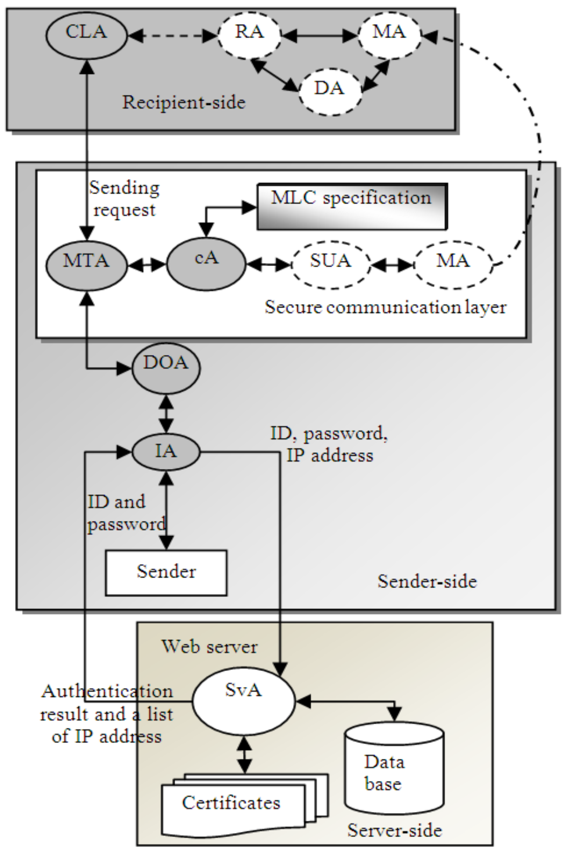

Fig. 5: The proposed MAgSeM

Each agent is skilled to perform certain specific tasks. An agent can be extended or instantiated by using the extensibility characteristic when its skill is required for the system to achieve the overall goal. For example, SUA is instantiated only when it is time to apply the desired cryptography protocols. DA is instantiated only when the condition is met to decrypt messages.

Agents in MAgSeM handle and automate the security processes with minimal intervention from the user. A mobile agent is used to carry secure data to the recipient. Mobile agents are suitable for a situation where the agent has only partial resources at the home platform and the rest of the resources are located at the recipient side. In our case, the mobile agent requires the private key of the recipient, which is only available at the recipient's host to decrypt the data. Mobile agents are robust, in a sense that if the destination platform is shut down while the agent is still there, the agent can take necessary actions such as migrating back or terminating its activities (Lange and Mitsuru, 1999). It can send a notice to the home platform about its situation and terminate if required.

Control over Data by Sender: MAgSeM focuses on a control mechanism on how a sender can securely transfer data to a recipient while *maintaining control* over the data. The 'maintaining control' over the data can be described as:

- If the message carried by the sender's mobile agent is seized by an attacker, the attacker still cannot recover the plaintext
- The recipient or any other third party does not need to know the details of the decryption processes to recover the plaintext

One way for the sender to gain control over the data, is to keep part of the requirements for the decryption process a secret, such as part of the agent's code, or parameters used for decryption. A symmetric key, K is used to encrypt the plaintext. This key and the information about the key (which is stored in the MLC specification), are kept with the sender until the mobile agent which has moved to the recipient's host needs it.

MAgSeM implements the control mechanism using the mobility and the extensibility of the agents. Two symmetric keys (K1 and K2) are used in the security processes. A sender agent uses K1 to encrypt plaintext to get a ciphertext. K2 is used to encrypt DA's code, which has the ability to decrypt the ciphertext.

A token, which is an encrypted random number, is carried by the mobile agent to the recipient's host. It is used as a 'phone home' mechanism (Grimley and Monroe, 1999), where the agent sends the token back to the sender. This is a way for the agent to tell the sender that it wants the information kept at the sender's side for the decryption processes.

Different agent's actions: We discuss the agent actions for MAgSeM's system. The following symbols will be used throughout this chapter to explain the security processes:

- Public and Private keys of the recipient: (pubKr, privKr)
- Public and Private keys of the sender: (pubKs, privKs)
- Symmetric keys: K1, K2
- Disposable secret and public key: (Ks, Kp)
- Plaintext: P
- Hash of P: H(P)
- Ciphertext: C
- Signature: S
- Agent's code: Cd
- Hash of Cd: H(Cd)
- A random number Rand
- Token: T
- The information extracted from the MLC Specification: mlc

An agent's code, which has the functionality to decrypt a ciphertext, is labelled as *Cd*, in which, when executed, Cd becomes DA.

Interface Agent (IA): IA provides Graphical User Interfaces (GUI) for authentication and message editing purposes. IA takes the user ID and password for authentication. When the user has finished writing a message, IA:

- Retrieves the recipient(s) names and the message from the interface
- Sends INFORM and both information to DOA and waits its confirmation of received message
- Waits for INFORM from MTA and displays a message to user regarding success or failure of the sent message

Data Organizer Agent (DOA): When DOA receives a message from IA, it does the following:

- Splits the message into Recipient(s) and the actual message
- Saves the actual message into a file (plaintext), which will later be retrieved by MTA
- Sends INFORM and the recipient(s) name to MTA and then to IA

Multi-tasking Agent (MTA): MTA receives the recipient(s) name from DOA and for each recipient, it does the handshaking process:

- Sends REQUEST to each recipient's CLA to send a message
- CLA will send an INFORM containing 'Agree' (or 'Reject'), a name of available RA and the agreed mlc for K2, which is shared between the sender and recipient
- Sends INFORM, with the recipients' address, RA's name and mlc specification for K2 to cA
- Any undelivered message notification will be handled
- Sends INFORM to DOA and then to IA regarding success or failure of sending the message

Communication Listener Agent (CLA): CLA listens to any incoming request to send message from other MTAs. When a new request is received:

- CLA checks the number of available RAs used at the moment. If the numbers of RAs are less than the maximum numbers allowed at a time (a maximum number of connections at a time can be set to control CPU processing power), then RA accepts the request with an 'Agree' answer, otherwise a 'Reject'
- Calculates com_layer for the communication and determine mlc from the MLC specification for K2. For Layer 2 communication on mobile device, the sender agent who makes the request will tell CLA, that it is using mobile device in the ACL message
- Creates an instance of RA, which will be responsible for entertaining any incoming MA carrying the sender's message
- Sends INFORM with RA's name and mlc to the MTA that made the request if the answer is 'Agree', otherwise sends 'Reject'

Crypto Agent (cA): When cA receives a message from MTA, it will do the following:

- Determines com_layer between sender and recipient based on their L_0s
- Based on com_layer, cA chooses mlc for K1 from the MLC specifications
- creates an instance of SUA to prepare the plaintext using cryptography protocols
- When creating SUA, the parameters that are passed to SUA are the RA's name and address, mlcs for K1 and K2, storage that keeps the private key and the recipient's certificate

- Monitors SUAs and the flow of the message, whether, the message is delivered properly and if not, give INFORM on the undelivered address to MTA

SetUp Agent (SUA): SUA takes the recipient's certificate and extracts pubKr. Then, it calculates com_layer for the sender and recipient's communication to obtain mlc. Then it:

- Generates two symmetric keys (K1, K2) according to mlc
- Encrypts P with K1 to produce a ciphertext C = E(P)K1
- Generates Rand and encrypts it with K1 to produce T that will be carried by MA. K1 is kept until T is received. T = E(Rand)K1
- Generates disposable (Ks, Kp). After T is received, Ks is used to encrypt the information that is kept for decryption processes. The corresponding Kp will be embedded in Cd and sent to the recipient's host to for decryption. The generation of (Kp, Ks) is one time per communication session. They will be disposed once the communication session is over, to avoid any third party from using Kp (Kp can be retrieved from the Recipient's host) in the next communication sessions.
- Take Cd and create a Java archive (.jar) file
- Signs the.jar file (Cd) with privKs to produce a signature, S, which is used to verify that Cd is from the sender. S = E(Cd)privKs
- Encrypts Cd, S and T with K2 to produce Ciphercode. Ciphercode = E(Cd, S, T)K2
- To allow only RA to retrieve K2, it is encrypted with pubKr together with H(Cd) to produce Cipherkey. Cipherkey = E(K2, H(Cd))pubKr
- Saves C, Ciphercode and Cipherkey in a file. Establishes SSL connection if necessary for channel security.
- Having finished preparing the message, SUA creates an instance of MA to carry the message to the recipient's host.
- Waits for T from MA. Once it is received, produce hashKey, which is the information to be given to MA that contains H(P), K1 and mlc. hashKey = E(K1,mlc,H(P))Ks

Mobile Agent (MA): MA carries the message to the recipient's host and there it communicates with RA.

- MA sends REQUEST to process the message
- Receives INFORM from RA indicating that both Cd and S are 'Valid'/'Invalid'. If 'Invalid' message is received, INFORM SUA and terminates
- If both are valid, MA sends REQUEST to sign T and sends the signed T back to SUA
- Receives hashKey from SUA and un-jarred Cd
- Send REQUEST to execute Cd
- hashKey and ciphertext are passed to Cd for the decryption processes

Receiver Agent (RA): RA will be in charged of communicating with MA and *Cd* in the process of decrypting a message:

- Waits for any request to process messages from MA
- Once received, the message is split into Ciphertext, Ciphercode and Cipherkey
- Gets privKr to decrypt Cipherkey D (Cipherkey)privKr = K2, H(Cd)
- Use K2 to decrypt Ciphercode D (Ciphercode) K2 = T, S, Cd
- Both S and Cd will be verified:
  - Validate S against Cd using the sender's pubKs
  - Recalculate H(Cd) from Cd in 4 and compare it with H(Cd) in 3
- If both S and H(Cd) are valid, sends INFORM to MA. If one or both are invalid, send a report to MA and abort current process.
- Sign T, when a request is made from MA.
- When the plaintext P and H(P) are received, recalculate H(P) and check if P is tampered.
- Sends INFORM to Cd whether P is 'Valid'/'Invalid'
- If P is valid, notify the recipient.

Decrypt Agent (DA): Once Cd is executed, it is called DA:

- DA decrypts hashKey using Kp D(hashKey)Kp = H(P), K1, mlc
- Loads and recreates K1 with mlc to decrypt the ciphertext, C
- Decrypts C to get P. D(C)K1 = P
- Sends INFORM to RA about P and H(P), so that RA can recalculate the hash and validate the plaintext.
- Terminates itself.

DISCUSSION

MAgSeM uses two secret keys (K1 and K2), with their specifications are derived from MLC. K1 is kept secret at the sender's side and by doing this; the sender has the advantages of gaining control over the data carried by MA. A recipient or any third party does not need to know the details of K1. Even though an attacker could intercept hashKey that came from Sender Agent, the attacker still cannot recover the plaintext as the key to decrypt hashKey, which is Kp, is at the recipient's host. In addition, Kp is disposable, which is created only once per every communication. Kp and the corresponding Ks will be removed once a communication session is terminated.

Another advantage for sender is that, when the token is received and validated, SUA knows that MA has been correctly executed at the recipient's host, because the correct token that has been recovered means that MA has been given the correct resources at the recipient's platform (which in this case the recipient's private key). Thus, the access to the resources at the recipient's host is not denied. In this case, RA must provide its private key to decrypt Cipherkey and sign T.

For RA on the other hand, it does not have to be burdened with the details of the decryption process of plaintext. RA is only required to provide its privKr and authenticate MA. It can verify that MA is indeed comes from the sender's host by checking the signature and the integrity of the agent's code (by verifying both S and H(Cd)). If both are valid, then the message and agent are indeed come from the trusted sender. In addition, it can check whether the plaintext is not tampered by calculating a new hash code and comparing it with the one received from the agent.

'INFORM' or 'REQUEST' is used to send messages among the agent as a form of communications among them. A message contains INFORM may be the partial results, which is needed by other agents to advance to their next actions to complete the overall system goal.

CONCLUSION

In this study we have presented the characteristics of MAS to develop MAgSeM. The desired characteristics include autonomy, mobility, extendible and interactive. These characteristics help in developing a system with less intervention from the user, where the security processes can be automated. MAgSeM employs a control mechanism that uses two secret keys and keeping one of them a secret a sender's side. This gives the sender a form of control over the data as a mechanism to protect the information. We use e-health domain as our problem domain and we conclude that MAgSeM is can also be used in other domains, where the users are distributed.

REFERENCES

Bellifemine, F.L., G. Caire and D. Greenwood, 2007. Developing Multi-Agent Systems with JADE. 1st Edn., John Wiley and Sons, Chichester, ISBN: 0470057475, pp: 286.

Braun, P. and W. Rossak, 2005. Mobile Agents: Basic Concepts, Mobility Models and the Tracy Toolkit. 1st Edn., Morgan Kaufmann Publishers Inc., San Francisco, CA., ISBN: 1558608176, pp: 441.

Debenham, J., 1999. An adaptive, maintainable, extensible process agent. Database Expert Syst. Appli. Lecture Notes Comput. Sci., 1677: 808-808. DOI: 10.1007/3-540-48309-8_59

Dierks, T. and C. Allen, 1999. The TLS Protocol Version 1.0. Internet Engineering Task Force (IETF).

Douligeris, C. and D.N. Serpanos, 2007. Network Security: Current Status and Future Directions. 1st Edn., John Wiley and Sons, Piscataway, NJ., ISBN: 0471703559, pp: 572.

Durfee, E.H., Lesser, V.R. and D.D. Corkill, 1987. Coherent cooperation among communicating problem solvers. IEEE Trans. Comput., 36: 1275-1291. DOI: 10.1109/TC.1987.5009468

Grimley, M.J. and B.D. Monroe, 1999. Protecting the integrity of agents: An exploration into letting agents loose in an unpredictable world. ACM, 5: 10-17. DOI: 10.1145/331648.331655

Huhns, M.N. and L. Gasser, 1989. Distributed Artificial Intelligence. 1st Edn., Pitman, London, ISBN: 0273088106, pp: 520.

Jennings, N.R., 2000. On agent-based software engineering. Artific. Intel., 117: 277-296. DOI: 10.1016/S0004-3702(99)00107-1

Lange, D.B. and O. Mitsuru, 1999. Seven good reasons for mobile agents. Mag. Commun. ACM, 42: 88-89. DOI: 10.1145/295685.298136

Liu, Q., S. Lu, Y. Hong, L. Wang and R. Dssouli, 2008. Securing telehealth applications in a web-based e-health portal. Proceedings of the 3rd International Conference on Availability, Reliability and Security, Mar. 4-7, IEEE Xplore Press, Barcelona, pp: 3-9. DOI: 10.1109/ARES.2008.9

Luck, M.M., 2001. Multi-Agent Systems and Applications. 1st Edn., Springer-Verlag, New York, pp: 118-149. ISBN: 3540423125, pp: 435.

Mahanti, A. and A. Bagchi, 1985. AND/OR graph heuristic search methods. J. ACM, 32: 28-51. DOI: 10.1145/2455.2459

Nwana, H.S., L. Lee and N.R. Jennings, 1996. Co-ordination in software agent systems. BT Technol. J., 14: 79-88.

Pardoe, T.D. and G. Snyder, 2005. Network Security. 1st Edn., Cengage Learning, Clifton Park, New York ISBN: 1401882145, pp: 485.

Shoham, Y. and M. Tennenholtz, 1992. On the synthesis of useful social laws for artificial agent societies. Proceeding of the Tenth National Conference on Artificial Intelligence, (AAAI' 92), John Wiley and Sons Ltd., USA., pp: 276-276.

Singh, M.P., 1998. Readings in Agents. 1st Edn., Morgan Kaufmann, San Francisco, Calif., ISBN: 1558604952, pp: 523.

SSH, 2003. SSH Secure Shell for Servers Version 3.2.9 Administrator's Guide. SSH Communication Security Corp.

Sulaiman, R., D. Sharma, W. Ma and D. Tran, 2007. A Multi-agent security framework for e-health services. Knowl. Intell. Inform. Eng. Syst., 4693: 547-554. DOI: 10.1007/978-3-540-74827-4_69

Wooldridge, M.J., 2002. An Introduction to Multiagent Systems. 1st Edn., John Wiley and Sons, West Sussex, England, ISBN-13: 9780471496915, pp: 348.