Design and Implementation of Inter-operable and Secure Agent Migration Protocol

Shakir-Ullah Shah[1], Jamil Ahmad[2], and Najeeb-ur-Rehman[3]
[1]Department of Computing and Technology, Iqra University Islamabad, Pakistan
[2]Department of Computer Science, University of Science and Technology (KUST), Pakistan
[3]Department of Computer Science, University of Gujrat, Pakistan

Abstract: *Mobile agent technology is an active research topic and has found its uses in various diverse areas ranging from simple personal assistance to complex distributed big data systems. Its usage permits offline and autonomous execution as compared to classical distributed systems. The free roaming nature of agents makes it prone to several security threats during its transit state, with an added overhead in its interoperability among different types of platforms. To address these problems, both software and hardware based approaches have been proposed to ensure protection at various transit points. However, these approaches do not ensure interoperability and protection to agents during transit over a channel, simultaneously. In this regard, an agent requires a trustworthy, interoperable, and adaptive protocol for secure migration. In this paper, to answer these research issues, we first analyse security flaws in existing agent protection frameworks. Second, we implemented a novel migration architecture which is: 1) fully inter-operable compliance to the Foundation for Intelligent Physical Agents (FIPA) and 2) trustworthy based on Computing Trusted Platform Module (TPM). The proposed approach is validated by testing on software TPM of IBM, JSR321, and jTPMTools as TPM and Trusted Computing Software Stack (TSS) interfaces, JADE-agent framework and 7Mobility Service (JIPMS). Validation is also performed on systems bearing physical TPM-chips. Moreover, some packages of JIPMS are also modified by embedding our proposed approach into their functions. Our performance results show that our approach merely adds an execution overhead during the binding and unbinding phases.*

Keywords: *Information Security, Multi-Agent Systems, Inter-Platform Agent Mobility, JADE, Trusted Computing.*

Received October 9, 2018; accepted February 24, 2019
https://doi.org/10.34028/iajit/17/4/4

1. Introduction

Mobile agent oriented technology is quite popularly used in e-commerce [35], search engines [36, 46], web-crawlers [33], scheduling problems [22, 30], parallel computing [24], health monitoring [39], network administration [34] and many other areas [41]. The mobility nature of the agent makes it more valuable while moving from one platform to another to perform different tasks on behalf of its owner. Along the course of migration on open networks, an agent may have private and confidential data that can be shared with only intended hosts or other agents. Maintaing trust in open networks is an open challenge and it is not a viable solution to provide a secure channel to each and every agent. Other problem with the agent is that when it visits different hosts, then a host may alter its secret data or even whole code. In such case an agent will not remain trustworthy. So at such scales, security and interoperability are the major challenges that affect usability and reliability [37] of Multi-Agent Systems (MAS). Three levels of security are required for MAS; host, channel, and agent security. In an open channel, a mobile agent is prone to Active [4], man in the middle (MITM) [27], and phishing [25] attacks which decreases level of trust. In this work, trusted and interoperable agent migration protocol based Trusted Computing is proposed. The Trusted Computing Group (TCG) prescribes the standards to increase level of trust based on a tamper resistant cryptographic hardware known as a Trusted Platform Module (TPM) [42]. For a proof of concept, Java Agent Development (JADE); a popular multi-agent Foundation for Intelligent Physical Agents (FIPA) compliant development framework [13] is used with TPM. Trust models of FIPA and their brief general security mechanisms for agent based systems is provided in [32]. The intrinsic TPM key hierarchy strengthens the security of the framework such that no device other than the intended host and platform can unbind or decrypt a private part of a binding key. It works by storing the Storage Root Key (SRK) in a non-volatile memory inside the TPM so that the whole tree structure becomes secure. As JADE does not support inter platform mobility, JIPMS [10] is used for this purpose.

In the next section (section 2), required background details pertaining to FIPA, JADE, and TC is provides. The section covers a literature review to demonstrate the progress made so far. Section 3 provides proposed

solution and section 4 provides the implementation details of the proposed protocol. This is followed by results and discussion section 5. Section 6 provides conclusion and future directions.

2. Background and Literature Review

A number of security attacks in the literature about the domain of mobile agent frameworks have been discussed in [20, 36, 38, 44]. The simplest form is an agent to agent attack involving one malicious agent attacking another on any host (local or remote) [6]. A malicious agent can also initiate an attack on its hosting platform in order to gain privileged access to the host. In this case, the attack is directed from malicious agent to an agent platform. But the counterpart can also be true. As an example, malicious platform can terminate the migration process an agent and can also truncate the data gathered by it in the form of a colluded truncation attack [23, 26]. Another variation involves attacks directed from one agent platform to another one. Other categories involve eavesdropping, denial of service, masquerading, unauthorized access, copy and replay, alteration, and man-in-the- middle attacks [36].

From time to time [6, 21, 23], general solutions for integrity of agents and platforms are provided. Zhang and Yan [49] provides solution for the agent security by introducing freezing mode. The secure session key is managed in a secure session database. For this purpose, a new layer of Transport Guard Process (TGP) is merged with the protocol and it is responsible for an agent's migration along with an Agent Control Center (ACC). The agent freezes itself at source host, half freezes itself over the network and unfreezes itself at the destination host. At the time of freezing, credentials such as timestamp, sequence number (nonce for protecting from replay attack), information/message digest (SHA-1), and digital signatures for integrity of digest are stored in the database. Authors in [14] developed a FIPA compliant trial involving a secure and efficient code distribution service known as JADE-S (JADE Security Module) [7]. The model uses Java Authentication and Authorization Service (JAAS), Java Secure Socket Extension (JSSE) and Java Cryptography Extension (JCE). User's credentials are store in plain form in a text file. Some other deficiencies in this model are discussed in [45].

Some software based approaches are also proposed but they are not able to provide fool proof security [9]. For this reason, TCG has proposed hardware solutions in form of TPM. Examples of software based solutions is a JADE and TPM4Java [19] which based on Secure Migration Library for Agents (SecMiLiA) [28] that bears functionality to address the malicious host problems. A downside of the library is that it works with a single key only at a time and as a result, it is not scalable. Code integrity is preserved using an approach used in [48]. In this approach, before migration, the integrity of the code is measured by taking the hash using SHA-1. At receiving side again hash is taken again and recorded. The code is allowed to resume for execution only after insuring the agent's integrity. Wu *et al*. [47] proposed different approach for code integrity. In this approach the whole structure is encrypted using public part of an Attestation Identity Key (AIK) of a destination host. A procedure known as Sealing and Unsealing is provided by TPM and used and discussed in [29]. The procedure is usable on local machines only and is strict in nature as it is locality dependent. A similar approach that guarantees integrity and confidentiality is also proposed in [1, 5] where hosts are authenticated using Public Key Infrastructures (PKI). A mobile agent-based crawler that promises to protect agents from malicious hosts is proposed in [43]. A self-reliant mobile code proposed in [40] promised to protect the agents selfishly. Symmetric keys are communicated in a secure way with digital signature. AI based approaches for protecting agents has been discussed in [8]. A robust and fault tolerant approach has been proposed in [31] that is based on security of code in a sedentary agents.

Interoperability in mobile agents can be handled at different levels such as executable code, platforms, and communication protocols [3]. The most common approach for dealing with interoperability is carried out at the application level. Our research focuses on interoperability issues arising due to migration, and for that the focus is on the communication protocols. A specification for FIPA compliant communication protocols utilizing Agent Communication Language (ACL) is discussed in [2, 16, 17]. For mobility, the Simple Mobility Protocol and Full Mobility Protocol stacks are used and described in [18].

The migration process as per FIPA standards requires three steps. The first step is the negotiation phase, where source and remote platforms communicate via ACL messages using the FIPA's Request Interaction Protocol (RIP). In the second step, configuration phase, the source and remote platforms share configurations and parameters. Lastly, in the execution phase, the selected transfer protocol is used for migration [12]. To achieve interoperability, mobility is placed under the Pre-Transfer, Transfer, and Post-Transfer protocols [13, 15] illustrated. In our implemented protocol, the same steps are used for mobility as described in [3]. Protocols which are used for transferring agent from one host to another are the push transfer, pull transfer, push ccache transfer, on-demand transfer, and fragment transfer protocols [12]. The proposed protocols discussed so far pertain to homogeneous platforms. For heterogeneous hosts, a detailed is in [11].

3. Proposed Solution

A new secure, flexible, and fast protocol for agent migration between a source and destination is proposed in this section using the Trusted Computing. The protocol uses symmetric keys, nonce, and timestamps to keep off replay attacks. Though the key management is out of the scope of this work but only an abstract idea is given for the sack of understanding. The Binding key Certification Authority (BKCA) is responsible for the management of RSA Binding Key certificates. The migration model of the process is designed as a layered architecture and is illustrated in Figure 1.

The migration model of the process is designed as a layered architecture and is illustrated in Figure 2. Agent Management System (AMS) communicate with a remote platform for the migration of an agent. It performs all requests and proceeds to the next mobility steps. FIPA Request Interaction Protocol (RIP) is the default migration protocol. All requests of an agent pertaining to migration are served. Similarly, it is responsible for serving all requests communicated by the Trusted Agent Migration Manager (TAMM).

The proposed agent, TAMM, acts like an interface and provides security for trusted T-move. The binding process involves TPM keys creation, return information about TPM status, return TCG compliant unique ID's labelled by manufacture, configure block size for both symmetric and asymmetric cryptography. At SH, when it receives public part of binding key, then it encrypts the agent and forwards it to AMS. AMS at the destination sends bound agent to TAMM to unbind.

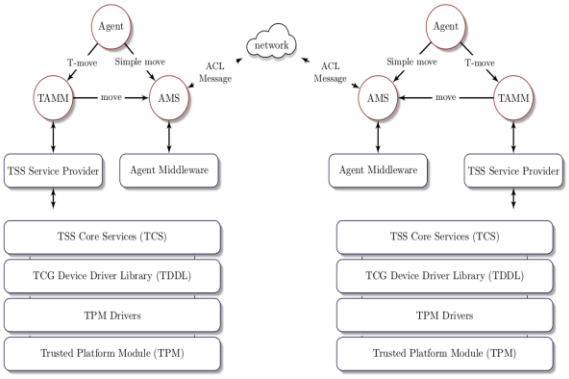

Figure 1. Layered architecture of trusted migration protocol.

Agent is at the top of the trusted migration protocol which is an entity that wants to move from SH to DH. For migration, an agent has two options; First being requesting the AMS to process the case via JIPMS mobility procedure, and the second being making the request to TAMM to process the case under TC specifications and FIPA standards. The former is termed as simple move and allows greater degree of flexibility, while the latter is termed as T-move and allows greater degree of security.

The Agent Middleware provides support and functional environment just like a standard platform for any real implementation of a multi-agent system. This is done by having support for the agent's entire life-cycle along with all expected transition states. The Trusted Computing Software Stack Layout is composed of TCG specified components for the functional environment. These components include the Trusted Platform Module (TPM), TPM Drivers, TPM Device Driver Library (TDDL), TSS Core Service (TCS), TSS Service Provider (TSP), and RPC/SOAP Communications. The TPM Drivers are provided by TPM manufacturers and support all required operations of the TPM. These drivers are loaded into the kernel as modules. The TDDL provides a standard interface to communicate with TPM and also provides support for switching between user and kernel mode. The TCS lies in user space and provides support for sophisticated tasks like key management efficiently. The TSP is the top most module in TSS layered architecture and provides an attractive object oriented interface to the applications. These applications then use an interface termed as the TSS Service Provider Interface (TSPI). The TSP executes in user-space while the TCS lies in kernel-space. For inter-communication, RPC/SOAP is used, which provides services like marshalling and un-marshalling. The TPM's protocol does not rely on security properties of the data transfer and assumes that all parts of the channel as un-trusted.

The proposed protocol follows a multi-protocol based setup which consists of Pre-Transfer, Transfer, and Post- Transfer protocol. The Pre-Transfer protocol comprises of a negotiation and configuration phase. The Transfer protocol comprises of the binding and transfer phase. The Post-Transfer protocol consists of resumption phase meant for resuming execution of an agent. Each of these phases are discussed in the coming sections.

When a AMS receives trusted migration request from a Mobile Agent (MA), see Figure 2, from its Source Host (SH) to a Destination Host (DH). Then steps given below are performed:

1. SH will forward migration request to DH.
2. Let DH is agree and responds to initiator.
3. SH bind the agent before t-move which consist further steps.
   a) First of It search the public key of DH in its local keys-repository, if fails then
   b) It request to Binding Key Certification Authority (BKCA) for it.
   c) BKCA tries to fetch it in its local repository. In success it returns, if fails.
   d) BKCS request to DH and return it SH.
4. SH binds the agent using received key.
5. DH unbind the agent using its own private part of binding key. DH informs SH about the successful power-up of agent.

The proposed protocol follows a layered architecture. The architecture is further divided into different phases. These phases consists of Pre-Transfer, Transfer, and Post-Transfer protocol. The Pre-Transfer protocol comprises of a negotiation and configuration phase. The Transfer protocol comprises of the binding and transfer phase. The Post-Transfer protocol consists of resumption phase meant for resuming execution of an agent. Each of these phases are discussed in the coming sections.

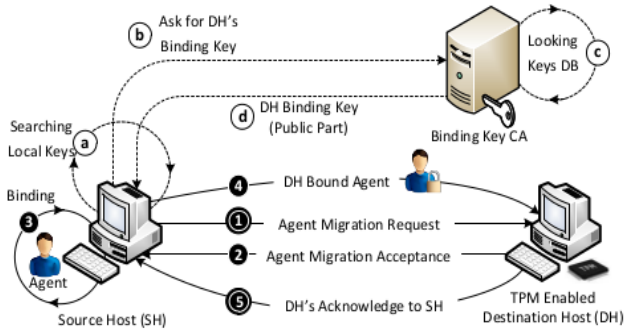

Figure 2. Overview of protocol.

3.1. Negotiation Phase

Negotiation is the first phase of the proposed model. This phase starts with a request of T-Move from an agent to the TAMM at SH ($TAMM_S$). The $TAMM_S$ forwards the move request to AMS_S which communicates with the destination AMS_D to get a decision for migration. Upon receipt of confirmation from AMS_S, $TAMM_S$ asks about the physical presence of TPM via AMS_D. AMS_D submit the query to $TAMM_D$ via TSP. As an inquirer, TSP finds whether TPM is enabled and activated, and replies with the status of this query. The $TAMM_D$ responds to its AMS about the TPM state returned via the TSP. This information is also delegated back to the SH. Both SH and DH try to agree upon a binding protocol for trusted move. Upon agreement, they select the required transfer protocol for the execution phase. Types of transfer protocols that can be selected are the Push, Pull, Push Cache, On-Demand, and Fragment Transfer Protocols. The entire process is open so that they can agree upon a common protocol with mutual understanding.

3.2. Configuration Phase

Configuration Phase the next phase of the proposed mode. The configuration phase receives all data acquired by SH and DH in the negotiation phase which includes TPM related information, binding key of DH. In T-Move the AMS_S will make a request AMS_D to get proof and status of DH TPM. AMS_D forwards the query to $TAMM_D$, then to TSP. TSP responds the proof and status of TPM to $TAMM_D$. The $TAMM_D$ forwards the response to AMS_S via AMS_D which in turn inform the $TAMM_S$. Now $TAMM_S$ is able to start T-Move. The $TAMM_S$ will try to locate the binding key at the local host. In case of failure the, the $TAMM_S$ will contact the BKCA. The BKCA will share only the public part of binding key.

3.3. Binding Phase

The core phase of this work is the binding phase which migrates the agent using a trusted T-Move by a TAMM. After getting the public of the binding key of DH, $TAMM_S$ binds the agent with DH according to TCG specification. Two different approaches were followed for the better results. In one approach the whole agent was bind. In another approach the agent was encrypted with AES and the key to decrypt was bind with DH. In either approach, the encrypted bounding agent is ready for T-Move. The detail of each approach is given in section 4.

3.4. Transfer Phase

This the last phase of the proposed model. In the transfer phase, agent will move to the DH using desired transfer protocol selected and agreed upon in the navigation and configuration phase by both parties i.e., SH and DH. As a result of the binding phase, the agent is already bind with DH. AMS_S will forward the bind agent to AMS_D which will make a request to TSP via $TAMM_D$ for unbinding. TSP will request TPM for this process. The TPM will unbind the agent using corresponding private part of the binding key along with UUID. The TSP then sends the request back to TAMM who in turn again forwards it back to the AMS as an unbound agent. The AMS then creates an agent thread and launches all required configurations. Finally, the destination AMS sends an acknowledgement to the source AMS containing status of the successful unbinding.

3.5. Execution Resumption Phase

In Execution Resumption Phase, the unbind agent will resume its execution based on its previous code, data and state. No changes were required for this phase.

4. Implementation

This section provides fine grained details for the implementation of proposed solution. In order to deploy the proposed solution, the testbed environment is created in two different ways. In first approach implementation is done on a Dell Latitude E6410 bearing a Broadcom TPM. In another approach, to confirm heterogeneity, the implementation has been tested on virtual TPM developed by IBM on Ubuntu 16.04. On software and hardware TPM enabled platforms, jTSS, jTpmTools and JADE 4.2 were configured. As JADE does not support inter-platform mobility, therefore JIMPS was also installed on both platforms.

Let E and D shows encryption and decryption algorithm respectively. PRSH shows private part of the binding of source host while PUDH shows public part of the binding key of the destination host.

In the first approach the data of the agent is encrypted with the PR_{SH} and then again encrypted with PR_{DH} to secure the data and avoid repudiation as shown in Equation (1).

$$E\ (E\ (Data;\ PR_{SH})\ ;\ PU_{DH}) \quad (1)$$

The public part of the binding key is extracted as a Key Binary Large Object (key-BLOB) using code outlined in algorithm 1.

```
Algorithm 1:Extract Public part of Binding-Key
#Require: UUIDRand, pubKeyFileName
#Ensure: Binding Key
contxt = TPMContxt.Instance( )
 contxt.connect(null)
kmmg = contxt.getKeyManager()
srk = kmng.LoadStorageRootKey(Secret.WELL KNOWN
SECRET)
bnd = kmng.createBindingKey(srk,Sec.WELL KNOWN SEC,
Sec.WELL KNOWNSEC, f, f, t, 2048, null)
kmng.storeTPMKey(srk,bnd,UUIDRand)
bndpub1= bnd.getPublicKey()
bndpub = bndpub1.Encoded()
writeFile(pubKeyFileName, bndpub)
```

When SH gets PR_{DH}, then SH binds the agent for T-move, code required to bind the data is listed in algorithm 2. When DH receives the bind agent then it unbind the agent. The code of unbinding is listed in algorithm 3.

In the second approach, the data of the agent is encrypted using AES encryption. The step by step procedure is shown in algorithm 4.

```
Algorithm 2: Agent Binding
#Require: byteData and pubKeyFileName
#Ensure: bData
rBndr = RemoteBinderImpl()
pubkbyt = readFile(pubKeyFileName)
kF = KeyFactory.getInstance("RSA")
publicKeySpec = X509EncodedKeySpec(pubk byt)
pu=(RSAPublicKey)kF.genPublic(publicKeySpec)
inBlockSize = 209
 outBlockSize = 256
DataIn = null
DataOut = null
numBlocks = (byteData.length / inBlockSize)
if byteData.length % inBlockSize != 0
{
   numBlocks++
}
 if numBlocks != 0
{
   numBlocks = 1
}
if bData == numBlocks *outBlockSize
{
   Arrays.fill(boundData, (byte) 0) }
i = 0
for i < numBlocks
{
DataIn = inBlockSize
DataOut = outBlockSize
DataOut= DataOut.fill( (byte) 0)
bData=, i * inBlockSize, DataIn,inBlockSize
DataOut = rBinder.bind[rawDataIn, pu]
DataOut=bData,i*outBlockSize,outputBlockSize
i = i+1
}
remaining = bData.length % inBlockSize
DataIn = byte[remaining]
DataIn= DataIn.fill, (byte) 0
DataOut = outBlockSize
DataOut =DataOut.fill, (byte) 0
bData=i * inBlockSize, DataIn, 0,remaining
DataOut =rBndr.bind(DataIn,pu)
DataOut=0, bData, i * outBlockSize,outBlockSize
return bData
```

This secret key is bound with public portion of the binding key of DH and encrypted again using private portion of the SH's binding key to avoid repudiation. Now this appended with the encrypted data. To decrypt the cipher, the secret key is obtained from the private part of public key.When migrating an agent, the private key is with the DH where the agent is supposed to be migrated.

$$\begin{aligned} E\ (p,sk) &= c1 \\ E\ (E\ (sk,PRSH),PUDH) &= c2 \\ c1|c2 &= c \end{aligned} \quad (2)$$

The secret key is then decrypted via the private part of the RSA key as depicted in Equation (2).

```
Algorithm 3: Agent Unbinding
#Require: bData, UUIDRand
#Ensure: unbData
unbData = null
contxt = TPMContxt.getInstance( )
contxt.connect(null)
kmng = context.getKeyManager( )
srk = kmng.loadStorageRootKey(Secret.WELL KNOWN
SECRET)
bnd2=kmng.loadTPMKey(srk,
UUIDRand,Secret.WELL KNOWN SECRET)
bndr = BinderImpl(contxt)
inBlockSize = 256 /*for unbinding
outBlockSize = 209 /*For unbinding
DataIn = null
DataOut = null
nBlocks =Math.ceil(bData.length/ inBlockSize
unbData=unbData.fill, (byte) 0
for (i=0; to i < numBlocks )
{
   DataIn = inBlockSize
   DataIn= DataIn.fill, (byte) 0
   DataOut = outBlockSize
   DataOut=DataOut.fill((byte) 0)
   bData=i * inBlockSize, DataIn,0,inBlockSize
   DataOut = bndr.unbind(DataIn, bnd2)
   if (i = numBlocks - 1 then
      DataOut=DataOut,unbData,i*outBlockSize,
     DataOut.length
   }
   else
   {
     DataOut=DataOut,unbData,i*outBlockSize,
```

```
        outBlockSize
      }
    }
    return unbData
```

The secret key is then decrypted via the private part of the RSA key as depicted in Equation (2). After getting the secret key, the data is decrypted with this key, such that it is obtained in its original form as shown in Equation (3).

Algorithm 4: AES Encryption

```
#Require: plaintext
#Ensure: ciphertext
keySp=SecKeySpec(AESKey.getBytes( ),"AES")
ivSpec = IvParameterSpec(ivx.getBytes( )
cipher=Cypher(keySp,ivSpec, Cipher.ENCMODE)
raw = cipher.doFinal(plaintext)
ciphertext = Base64.encode(raw);
return ciphertext
```

After getting the secret key, the data is decrypted with this key, such that it is obtained in its original form as shown in Equation (3).

$$D(D(c_1, PU_{SH}), PR_{DH}) = sk$$
$$E(c_1, sk) = p \quad (3)$$

In algorithm 5, byte stream are encrypted using AES to generate cipher stream. Then first byte of cipher stream is decoded to UTF-8 from Base64. Now the cipher stream is for decryption. Finally, cipher stream is decrypted into plain stream.

Algorithm 5: AES Decription

```
Require: EncryptedText, AESKey
Ensure: DecryptedText
keySp = SecKeySpec(AESKey.getBytes(), "AES")
ivSpec = IvParameterSpec(ivx.getBytes())
cipher=Cypher(keySp,ivSpec, Cipher.DECMODE)
raw = Base64.decode(encryptedText)
decryptedText = cipher.doFinal(raw)
clear = String(decryptedText, "UTF8")
return decryptedText
```

5. Results and Discussion

Once the test-bed is deployed we then test the performance of it by taking readings and comparing it with different benchmarks. Binding time and unbinding time are the two parameters on which the performance of the test-bed is going to be analysed. To conduct the test, we take multiple agents which vary in sizes from each other. The sizes of agents are in the range of 0.25KB to 4096 KB. First we start from a small agent of size 256 bytes after that we increase the size of the agent by 2s power such as 512, 1024, 2048 and so on. Table 1 gives us the information of agent size in Bytes and Kilobytes (KBs). Also, it gives us the binding time and unbinding time with respect to the given agent.

Table 1. Time complexity using simple approach.

S.No.	Size (KB)	Binding Time (MS)	Unbinding Time (MS)
1	0.25	6.009	856.494
2	0.5	9.444	1085.715
3	1	13.416	1714.031
4	2	15.574	3091.546
5	4	25.743	5641.002
6	8	36.544	11170.354
7	16	79.676	21635.348
8	32	130.633	42448.617
9	64	245.002	84661.390
10	128	439.058	169117.530
11	256	862.117	337308.620
12	512	1677.559	674150.400
13	1024	3740.121	1349534.500
14	2048	8252.697	2698777.200
15	4096	14561.789	5399847.500

To visualize the correlation between binding time and unbinding time with respect to the size of agent we use a line chart. Which could typically bring ease in helping us to find patterns from the given results. X-Axis: On X-axis we have the size of an agent in Kilobytes (KBs) Y-Axis: On Y-axis we have time dimension which is given in Milliseconds. Figure 3 has a line chart that shows the relation of unbinding time in milliseconds with respect to the agent size. Like the binding timeline chart, the unbinding timeline chart has a solid blue line that represents unbinding time with respect to the agent size. The second dotted line shows the linear unbinding time of agent with different sizes. By looking at Figure 3 we have come to a conclusion it is not feasible to deploy this solution as the execution time of binding and unbinding increases exponentially with the increase of the agent size. Since encrypting and decrypting the whole agent during binding and unbinding gets computationally expensive with the increase in the size of the agent, therefore, the solution does not remain feasible anymore. So, to overcome this problem we incorporate Advanced Encryption Standard (AES) which is a symmetric key cryptographic technique. Incorporation of AES would work in a way that at the binding time of agent we would bind the public key of the source and attach the public key with an agent. Once the agent reaches the destination the agent is decrypted by combining the public and private together. So during the whole pipeline, we are encrypting the public keys instead of agents which would save a lot of overhead. Secondly, the perks of public key encryption are that TPM ensures that the agent would only decrypt on the destination platform which ensures the integrity of the information delivered. This process typically increases the decryption performance which reduces the time latency gap in binding and unbinding execution costs. To assess the improvements by incorporating AES to our proposed method we have a look at Table 2 numbers.

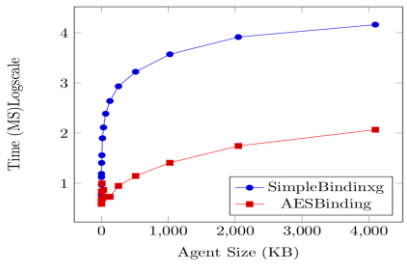

Figure 3. Binding execution time over agent size.

Table 2 shows the recorded results of binding and unbinding with support for AES. Considering the same cases, the binding takes 4 MS instead of 15 MS when the agent size is 2 Kb. Likewise, it takes only 8 MS instead of 862 MS when agent size is 256 Kb, and so on. Finally, if agent size is 4096 Kb, it is bound 117 MS instead of 14561 MS. Similarly, unbinding takes 565 MS instead of 3091 MS when the agent size is 2 Kb, 575 MS instead of 5 minutes when agent size is 256 Kb and so on until an improvement of 669 MS is noticed instead of 1.5 hours for an agent size of 4096 Kb.

Table 2. Time complexity using AES.

S.No.	Size (KB)	Binding Time (MS)	Unbinding Time (MS)
1	2.5	3.926	595.912
2	0.5	4.157	590.353
3	1	4.739	563.948
4	2	4.864	565.545
5	4	6.939	570.306
6	8	10.063	570.313
7	16	5.058	581.930
8	32	7.457	581.754
9	64	5.460	584.201
10	128	5.440	598.874
11	256	8.957	575.637
12	512	14.130	585.334
13	1024	25.859	601.556
14	2048	55.936	624.109
15	4096	117.981	669.442

Figure 4 shows efficient computational results of the proposed model of T-move of binding with AES. There are two lines, blue one shows the behavior of binding with AES Encryption and red one shows the result of unbinding using second approach.

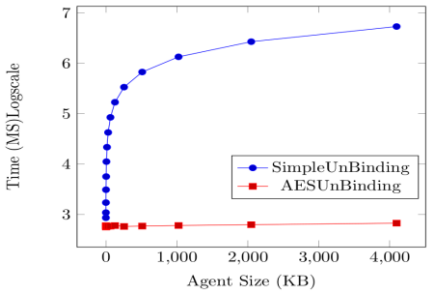

Figure 4 . Unbinding execution time over agent size.

6. Conclusions

The growing nature of the structured and unstructured date over Internet makes it very difficult to collect one's own required data. Mobile agents are especially useful to perform functions automatically including information extraction and dissemination. On the other hand, this expanded power and usage, is also causing security concerns. The goal of this research work is to give protection to mobile agents without any sort of compromises on the integrity of information. Also, the research work enables us to conduct secure transfer and exchange of information through mobile agents. The proposed secure model mainly consists of architectural flow, migration model and sequential execution for the exchange of information with ease. For trustworthy transfer, proposed novel trusted model is used to ensure secure migration of agents from one host to another. In order to ensure the effectiveness and efficiency in actual environment, Trusted Platform Module has been used. Computing Group (TCG) which is highly recommended by experts of the domain. Hieratical security of Trusted Platform Module (TPM) by TCG is used to enhance the security and to ensure that the binding keys are not made public other than the destination host. The full migration protocol developed by us has elastic architectures which enable mobile agents to migrate with ease. The proposed method provides the user with the flexibility of choosing the security protocol instead of imposing it on the user. Our methods also let the user choose between normal and trusted migration depending upon the user requirements. Securing the communication medium during the migration of agent is beyond the scope of this research. Therefore if the keys are extracted through the medium we cannot do anything about it currently. Also, our research does not share the public part of the binding key as it is beyond the scope of our work. Our methods provide a protected and reliable environment for mobile agent relocation, however, users can choose explicitly the mode of migration. However out method at the moment does not have a digital signature and we have a constant size of AES key length but this would be addressed by us in the future. Further improvements can be made and the network issue can be resolved during the migration by choosing the best transfer protocol for the agents. TO enhance elasticity and scalability of a multi-agent system in a trusted model we should incorporate TPM like a proxy setup.

References

[1] Aljawarneh S. and Alhaj A., "Testing and Evaluation of A Secure Integrity Measurement System (SIMS) for Remote Systems," *The*

International Arab Journal of Information Technology, vol. 9, no. 3, pp. 235-242, 2012.

[2] Ametller J., Robles S., and Borrell J., "Agent Migration over FIPA ACL Messages," *in Proceedings of International Workshop on Mobile Agents for Telecommunication Applications*, Berlin, pp. 210-219, 2003.

[3] Ametller-Esquerra J., Cucurull-Juan J., Martí R., Navarro G., and Robles S., "Enabling Mobile Agents Interoperability Through Fipa Standards," *in Proceedings of International Workshop on Cooperative Information Agents*, Berlin, pp. 388-401, 2006.

[4] Andress J., "The Basics of Information Security: Understanding the Fundamentals of Infosec in Theory and Practice," *Syngress*, 2014.

[5] Bellavista P., Corradi., A and Stefanelli C., "A Mobile Agent Infrastructure for The Mobility Support," *in Proceedings of the ACM Symposium on Applied Computing*, Como, pp. 539-546, 2000.

[6] Bijani S. and Robertson D., "A Review of Attacks and Security Approaches in Open Multi-Agent Systems," *Artificial Intelligence Review*, vol. 42, no. 4, pp. 607-636, 2014.

[7] Board J.A.D.E., Jade Security Guide. JADE-S Technical Report, 2005.

[8] Brahmi Z., Lini A., and Gammoudi M., "Mobile Agent Security Based on Artificial Immune System," *in Proceedings of International Joint Conference SOCO'14-CISIS'14-ICEUTE'14*, Cham, pp. 385-395, 2014.

[9] Challener D., Yoder K., Catherman R.., Safford D., and Van Doorn L., A Practical Guide to Trusted Computing Pearson Education, 2007.

[10] Cucurull J., "JADE Inter-Platform Mobility Service," Online Available and accessed from: <http://jipms.sourceforge.net>, 2018.

[11] Cucurull J., Martí R., Navarro-Arribas G., Robles S., and Borrell J., "Full Mobile Agent Interoperability in An IEEE-FIPA Context," *Journal of Systems and Software*, vol. 82, no. 12, pp. 1927-1940, 2009.

[12] Cucurull J., Martí R., Robles S., Navarro G., and Borrell J., "Agent Mobility Architecture Based on IEEE-FIPA Standards," *Computer Communications*, vol. 32, no. 4, pp. 712-729, 2009.

[13] Cucurull J., Marti R., Robles S., Borrell J., and Navarro G., "FIPA-Based Interoperable Agent Mobility Proposal," Technical Report, 2007.

[14] Cucurull J., Navarro-Arribas G., Martí R., Robles S., and Borrell J., "An Efficient and Secure Agent Code Distribution Service," *Journal of Software: Practice and Experience*, vol. 40, no. 4, pp. 363-386, 2010.

[15] Cucurull J., Overeinder B.J., Oey M., Borrell J., and Brazier F., "Abstract Software Migration Architecture Towards Agent Middleware Interoperability," *in Proceedings of the 2nd International Multiconference on Computer Science and Information Technology*, Poland, pp. 27-37, 2007.

[16] Dale J., and Lyell M., "Foundation for Intelligent Physical Agents," http://www.fipa.org/, Last Visited, 2019.

[17] Fipa A.C.L., "FIPA acl Message Structure Specification," http://www.fipa.org/, Last Visited, 2019.

[18] FIPA: FIPA agent management support for mobility specification, 2019.

[19] Hermanowski M. and Tews E., "Tpm4java". Currently only available through http://web. archive.org/web/20090510093615/http://tpm4java. datenzone. de/trac, Last Visited, 2018.

[20] Jolly P. and Batra S., "Security AGAINST Attacks and Malicious Code Execution in Mobile Agent Using IBF-CPABE Protocol," *Wireless Personal Communications*, vol. 107, no. 2, pp.1-15, 2019.

[21] Jung Y., Kim M., Masoumzadeh A., and Joshi J., "A Survey of Security Issue in Multi-Agent Systems," *Artificial Intelligence Review*, vol. 37, no. 3, pp. 239-260, 2012.

[22] Kori S. and Kakkasageri M., "Intelligent Agent Based Resource Scheduling in Wireless Sensor Networks," *in Proceedings of 10th International Conference on Computing, Communication and Networking Technologies ICCCNT*, Kanpur, pp. 1-7, 2019.

[23] Lei S., Liu J., and Xiao J., "A Novel Free-Roaming Mobile Agent Security Mechanism by Trusted Computing Technology," *in Proceedings of International Conference on Computer Science and Software Engineering*, Hubei, pp. 721-724, 2008.

[24] Liu J., Zhang S. and Yang J., "Characterizing Web Usage Regularities with Information Foraging Agents," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 5, pp. 566-584, 2004.

[25] Loy S., Brown S., and Tabibzadeh K., "South Carolina Department of Revenue: Mother of Government Dysfunction," *Journal of The International Academy for Case Studies*, vol. 20, no. 1, pp. 83-93, 2014.

[26] Marikkannu P., Murugesan R. and Purusothaman T., "AFDB Security Protocol Against Colluded Truncation Attack in Free Roaming Mobile Agent Environment," *in Proceedings of International Conference on Recent Trends in Information Technology*, Chennai, pp. 240-244, 2011.

[27] Mishra P., "Analysis of Mitm Attack in Secure Simple Pairing," *Journal of Global Research in Computer Science*, vol. 4, no. 2, pp. 42-45, 2013.

[28] Munoz A., Mana A., and Anton P., "A Solution Based on Cryptographic Hardware to Protect Agents," *in Proceedings of 13th International Conference on Network-Based Information Systems*, Takayama, pp. 400-407, 2010.

[29] Munoz A., Mana A., and Serrano D., "Protecting Agents from Malicious Hosts Using TPM," *International Journal on Computational Science and Applications*, vol. 6, no. 5, pp. 30-58, 2009.

[30] Neagu N., Dorer K., Greenwood D., and Calisti M., "LS/ATN: Reporting on A Successful Agent-Based Solution for Transport Logistics Optimization," *in IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications*, Prague, pp. 213-218, 2006.

[31] Ouardani A., Pierre S., and Boucheneb H., "A Security Protocol for Mobile Agents Based Upon The Cooperation of Sedentary Agents," *Journal of Network and Computer Applications*, vol. 30 no. 3, pp. 1228-1243, 2007.

[32] Poslad S., Charlton P., and Calisti M., "Specifying Standard Security Mechanisms in Multi-Agent Systems," *in Proceedings of Workshop on Deception, Fraud and Trust in Agent Societies*, Berlin, pp. 163-176, 2003.

[33] Thati P., Hao C., and Agha G., "Crawlets: Agents for High Performance Web Search Engines," *in Proceedings of International Conference on Mobile Agents*, Berlin, pp. 119-134, 2001.

[34] Pulter N., Nimis J., and Lockemann P., "Managing Contingencies in Timed Transportation Networks by Agent Technology," *in Proceedings of the Workshop on Artificial Intelligence and Logistics (AILog-2010) at the 19th European Conference on Artificial Intelligence*, Lisbon, 2010.

[35] Rahman S. and Bignall R., Internet Commerce and Software Agents: Cases, Technologies, and Opportunities, IGI Global, 2001.

[36] Rajeshwar B., Saravanan, A., Balaji R., Geetha, G., and Jayakumar, C., "Secure Information Retrieval Using Mobile Agent," *in Proceedings of the International Conference on Computing and Control Engineering*, pp. 12-13, 2012.

[37] Rocha Á., Correia A, Wilson T., and Stroetmann K., *Advances in Information Systems and Technologies*, Springer Science and Business Media, 2013.

[38] Shen Z. and Tong Q., "A Security Technology for Mobile Agent System Improved by Trusted Computing Platform," *in Proceedings of 9th International Conference on Hybrid Intelligent Systems*, Shenyang, pp. 46-50, 2009.

[39] Smarsly K., Law K., and Hartmann D., "Multiagent-Based Collaborative Framework for A Self-Managing Structural Health Monitoring System," *Journal of Computing in Civil Engineering*, vol. 26, no. 1, pp. 76-89, 2012.

[40] Srivastava S. and Nandi G., "Self-Reliant Mobile Code: A New Direction of Agent Security," *Journal of Network and Computer Applications*, vol. 37, pp. 62-75, 2014.

[41] Sun Y., Councill I., and Giles, C., "Botseer: An Automated Information System for Analyzing Web Robots," *in Proceedings of 8th International Conference on Web Engineering*, Yorktown Heights, pp. 108-114, 2008.

[42] Trusted Computing Group, "TCG Specification Architecture Overview, Specification Revision, Last Visited, 2018.

[43] Upadhyay V., Balwan J., and Shankar G., "A Security Approach for Mobile Agent Based Crawler," *in Proceedings of the 2nd International Conference on Computer Science, Engineering and Applications*m, pp. 119-123, New Delhi, 2012.

[44] Vigna G., "Mobile agents: Ten reasons for failure," *in Proceedings IEEE International Conference on Mobile Data Management*, Berkeley, pp. 298-299, 2004.

[45] Vila X., Schuster A., and Riera A., "Security for A Multi-Agent System Based on JADE". *Computers and Security*, vol. 26, no. 5, pp. 391-400, 2007.

[46] Wang M., Li Q., Lin Y., Li Y., and Zhou B., "A Personalized Metasearch Engine Based on Multi-agent System," *The International Arab Journal of Information Technology*, vol. 16, no. 6, pp. 978-987, 2019.

[47] Wu X., Shen Z., and Zhang H., "Secure Key Management of Mobile Agent System Using Tpm-Based Technology on Trusted Computing Platform," *in Proceedings of International Conference on Computer Science and Software Engineering*, Wuhan, pp. 1020-1023, 2008.

[48] Xian H. and Feng D., "Protecting Mobile Agents' Data Using Trusted Computing Technology," *Journal of Communication and Computer*, vol. 4, no. 3, pp. 44-57, 2007.

[49] Zhang W. and Yan X., "Agent Transport Security Based on Freezing Mode," *in Proceedings of International Conference on Communications and Intelligence Information Security*, United States, pp. 60-63, 2010.

Shakir-Ullah Shah has got MSCS from Foundation University Islamabad, Pakistan with a gold medal. He is working as a assistant professor at National University of Computer & Emerging Sciences Peshawar, Pakistan. His area of interest is information security in multi-agent systems.His previous research at MS level was about authentication factors and led to explore another authentication factor i.e. “something you process”. The main objective of his current research is to secure users’ credentials and to enhance the usability of authentication and authorization.

Jamil Ahmad He got PhD from King's College London, UK and MSC in Information Technology from University of Warwich, UK. His current research are in Artificial Intelligence, Artificial Neural Networks, Image processing and Machine learning. He worked as a vice chancellor of Iqra University, Islamabad, Pakistan, Abasyn University, Islamabad, Pakistan. Currently he is working as a voice chancellor of Kohat University, Pakistan.

Najeeb-ur-Rehman academic abilities can be perceived from the fact that he has completed his BS (CS), and MS (CS) degrees with Cum Laude honor from FAST-National University of Computer and Emerging Sciences (NUCES) and nominated as Gold Medalist. Even more, he is nominated for Dean’s List of Honors and Rector’s List of Honors several times. He is Gold Medalist in Software and Hardware IT Competition organized by DOST-KPK and attend numerous Computer Programming and IT Contests all over the Pakistan.