CHAPTER 4

MOBILE AGENT PLATFORM PROTECTION

4.1 INTRODUCTION

The Mobile Agent Platform is one of the most important parts of the Mobile Agent System. It provides all the resources required for the execution of a Mobile Agent. Hence, it is vital to protect the Mobile Agent platform from attackers. This work presents two mechanisms to protect the Mobile Agent Platform which are discussed below.

1. Dual Check-point Analysis
2. Improved Malicious Identifier with Tripmarker

4.2 DUAL CHECK POINT ANALYSIS AGAINST TAILGATING

The concept of Mobile Agents has revolutionized the world of computing by introducing mobility to the code and by reducing the traffic in the network. Due to this, the usage of Mobile Agents has been introduced in a number of real time applications. With the growth in technology, the growth of threats against the technology has also been found to be increasing. As a result of this, the trust over technology may get affected. However, with the growing threats, security mechanisms to overcome the threats have also been proposed by various experts related to the domain. Some of the major threats have been identified and solutions are proposed to overcome those. Some may not be identified easily, but could be identified only through keen observation. On such a context, with keen observation over Mobile Agent system failure it
was identified that an attack which had been used in the physical world has been introduced in the world of Mobile Agents. The attack that had been used was the Tailgating attack which has less significance in the domain of Information Security, since it is not electronically used widely. But now, it has been identified that it poses a major threat towards the security of Mobile Agents.

In this work, a technique named Dual Check-Point has been used to combat with the Tailgating attacks. Also, this technique incorporates the fragmentation and defragmentation of data in order to provide complete support to the Dual Check-point security technique. The main advantage of this technique is that the system, as a whole, could be controlled by the administrator. Also, this would ensure a secure Mobile Agent environment which is immune to Tailgating attacks.

4.3 BACKGROUND STUDY

Mobile Agents are piece of codes or modules which are capable of moving from one location to another in order to accomplish a task. The location may be a platform in some other host or a platform in the same host. The mobility of the code is considered to be advantageous which would help in the reduction of load in the network. A Mobile Agent is capable of collecting data from various sources. These collected data could be transferred to some other system which integrates and processes the data. The Mobile Agent itself is capable of processing the data it holds. The Mobile Agents are used in various applications related to a distributed system where transfer of information from one part of the system to another is a mandatory one. A Mobile Agent supports asynchronous and autonomous operation, allows dynamic and flexible adaptation to a changing environment, allows reduction of communication cost and allows encapsulation of protocols (Joachim Baumann 2000).
The Mobile Agent also exhibits the features such as autonomy, social behaviour, reactivity and proactivity which overcome the drawback of the client-server architecture (Peter Braun and Wilhelm Rossak 2005).

Tailgating is a security threat which is often encountered in the physical world. It involves a process in which a person, whether an employee or not, passes through a secure door without the knowledge of the person who has legitimate access to the particular secure door (Lance Hayden 2010, Harold Tipton and Micki Krause 2006).

The same case is applicable for Mobile Agents also. In the Mobile Agent system, a Mobile Agent may possess the access to a secure system. Some malicious Mobile Agents may enters the system along with a legitimate Mobile Agent.

In the Mobile Agent System, there is a possibility of tailgating attack in the distributed environment. The Digital Signature by Timestamp based Elliptic Curve Cryptography (ECC) is used to identify Unauthorized Access attack. ECC is one of the technologies used to generate the keys. Key size in ECC is 160 bits which is secure than RSA key size (1024 bits). Hence ECC gives lower memory requirement and greater execution speed. Akl et al (1983) proposed the cryptography mechanism for solving the problem of access control in a hierarchical manner. This hierarchy stores a large number of cryptographic keys. The disadvantage is that a large number of keys need to be stored by each user.

Volker and Mehrdad (1998) proposed a security scheme, which is based on a tree structure, to design a secure environment for Mobile Agents. The Mobile Agent structure can be broken into two main branches: static branch, the information stored in the agent should not be modified during the agent’s life time. The information about the agent includes security policy,
group classification, certificates and access control keys. Mutable branches
store the dynamic information that could be modified by the visited host on
the network. The larger the amount of duplicated keys and key-derivation
computations, the more inefficient the scheme becomes.

Naoya Torii et al (2000) describe the Elliptic Curve Cryptosystems
(ECC) as the next generation of public key cryptosystems with shorter key
length and provide equal security levels, implemented fast with less hardware.
It outlines ECC and then describes a typical digital signature algorithm. ECC
includes key distribution and encryption. The technology for generation of a
secure ECC is based on the computation of parameters such as the prime p,
Elliptic Curve E, base point G=(x,y) and the order of the curve. The security
of an ECC depends not only on the length of the key, but also on the Elliptic
Curve parameters. When the stronger attack encounters, security levels will
get changed.

Yu Fang Chung et al (2008) proposed a novel key management
method based on ECC and one-way hash function to solve dynamic access
problems in a user hierarchy. This scheme attempts to derive the secret key of
successors efficiently and non-redundantly. The Central Authority enables a
user to change the secret key conveniently. It uses the ECC which has a low
computational cost and smaller key size, in terms of both security and
efficiency. As the hierarchical process of updating a key becomes complex,
the procedure for altering secret keys and storage space becomes
inconvenient. The proposed method simplifies key generation and derivation
algorithms, efficiently and solves the dynamic access control problems,
ensures the users to alter their secret keys for security reasons, and resists
collusive attacks. The members access the data according to their ranks. The
members in higher-ranked security class can directly access the secret keys of
members in lower-ranked classes, but not vice versa.
A hierarchical structure of Mobile Agents is proposed by Chia-Hui Liu et al (2010) to overcome the drawbacks of Volker and Mehrdad’s scheme. Larger amount of duplicated keys and key-derivation computations makes the scheme to be inefficient. Elliptic curves are used to construct a key management for controlling the access of private keys. The characteristics of ID-based in bilinear pairing will be applied to build a hierarchical structure of Mobile Agents with the technology of key management. This reduces the amount of keys and decreases the key-derivation computations. The drawback is that the computation of public key is very high.

Tzer Long Chen et al (2010) uses the Elliptic curve cryptography to enhance security by providing shorter key length and efficient key management using date bound warrant. When the user decides to log off the system, the system must cancel the rights of the previously assigned key. To overcome this issue date constraint scheme has been proposed. In this work the proof of accuracy does not meet the time requirements.

Karnik et al (1999) propose the mechanism of AppendOnlyContainer for detecting the alteration of an agent’s data by individual malicious hosts. The concept of checksum is used to detect the alteration of the agent’s data. However, the mechanism does not identify the malicious hosts.

Vijil et al (2002) proposed the Extended AppendOnlyContainer mechanism. This mechanism also uses the concept of checksum to detect alteration in the agent’s data. The same technique is employed to detect the malicious hosts that collude with each other to delete the data of other hosts, for static as well as dynamic itineraries. The mechanism for authentication against unauthorized attack has not been discussed.
4.4 SCENARIO

While considering the tailgating attack, not much literature is available of the mode of attack or the nature of attack with respect to a Mobile Agent system. By our keen observation, we had found that the malicious Mobile Agent appends with the legitimate Mobile Agent either by attacking the Mobile Agent or by injecting the malicious code within the Mobile Agent. This may make it possible for the Mobile Agent system to allow the malicious agent within itself thereby allowing the entire system to be affected.

Consider a secure bank system. A legitimate Mobile Agent, which works on behalf of a user, carries the account number and the Personal Index Number within it. The attacker desires to access the account details or transfer the credits from the user’s account to his account. So, the attacker designs a Mobile Agent which would append with the legitimate Mobile Agent without disturbing the internal functionalities or the data within the agent.

When the legitimate agent, who carries the malicious Mobile Agent reaches the gateway of the secure bank system and gets access by using the account number and the Personal Index Number. The malicious Mobile Agent, as soon as it enters the secure bank system detaches from the legitimate Mobile Agent and starts functioning autonomously and gains access over the system. Further, it starts performing its malicious functionalities which may affect the entire system.

This work proposes a Mobile Agent system architecture which would protect the system from Tailgating attacks and maintain the stability of the system. The Quality of Protection is assured by modeling the system using the proposed architecture.
4.5 PROPOSED ARCHITECTURE

The proposed architecture to overcome the tailgating attack and proxy signing are shown below.

4.5.1 Tailgating Attack

The Secure Mobile Agent system proposed in this chapter contains the following components of which most of the components are integrated within the Agent Platform.

1. Authentication Table
2. Passage
3. Authentication Check
4. Agent Generator
5. Fragmentation
6. Integrity Verifier
7. Defragmentation

Secure Mobile Agent System Architecture is shown in Figure 4.1.
4.5.1 Authentication Table

The authentication table is a centralized database which contains details of the Mobile Agents generated in the system. The Mobile Agent system registers all the Mobile Agents created by it to the authentication table. Only those Mobile Agents which are registered with this table are allowed to access the gate. The authentication table is kept in a secure environment. The structure of the Authentication table is shown Figure 4.2.

<table>
<thead>
<tr>
<th>Agent ID</th>
<th>Host ID</th>
<th>Size (in kb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>001</td>
<td>25</td>
</tr>
<tr>
<td>025</td>
<td>003</td>
<td>29</td>
</tr>
<tr>
<td>023</td>
<td>002</td>
<td>32</td>
</tr>
<tr>
<td>027</td>
<td>001</td>
<td>31</td>
</tr>
</tbody>
</table>

Figure 4.2 Authentication Table
The Agent ID corresponds to the individual identity of a Mobile Agent. The Host ID is the identity of the Host from where the Mobile Agent was generated. The size corresponds to the total size of the Mobile Agent along with the data.

4.5.1.2 Passage

The passage plays a very important role in the Mobile Agent system. The passage uses the Dual Check-point entry scheme, which is very helpful in controlling Tailgating attacks. The passage has two gates namely the external gate and the internal gate. The passage contains a buffer of definite size which could be determined by the administrator. The Mobile Agents carrying the data must be of the size of this buffer. The process is discussed in the later part of this chapter. The passage is connected to the Authentication Check which checks whether a Mobile Agent is legitimate or not. The passage is just a locker which holds the Mobile Agent inside itself at the time of verification. When a Mobile Agent enters the passage, it is forced to be in a zero-execution state in order to prevent damages, in case if a malicious Mobile Agent tries to breach it.

4.5.1.3 Authentication Check

The Authentication Check is one of the most important components of the Mobile Agent System. It obtains authentication details from the central authentication table. It performs authentication checking at the external as well as internal gates of the passage. This also could instruct the passage to kill a Mobile Agent if it is found to be malicious.

4.5.1.4 Agent Generator

The agent generator is responsible for the creation of new Mobile Agents which would transmit data from one platform to another.
4.5.1.5 Fragmentation

The fragmentation module is one which makes a main contribution to the sizing of the Mobile Agents. When the size of the Mobile Agent in combination with the data is larger than the buffer size of the passage, the fragmentation module fragments the data and helps to maintain the stability of the system. The excess data is carried by new Mobile Agents generated by the Agent Generator. Those Mobile Agents do not perform any special task other than carrying the data. They get destroyed once they reach the destination and offer the data to the destination platform. The Fragmentation module sets a fragment bit and a fragmentation sequence number for the data that is being carried by the Mobile Agents.

In Figure 4.3, the format of the Mobile Agent with the data is shown. The notations or numbers given within the brackets indicate the maximum allowable size of the field in bits. AID refers to the Agent Identifier which uniquely distinguishes the Mobile Agent in the system. AID is an 8 bit field. HID refers to the Host Identifier which refers to the identity by which a host is distinguished in the entire system. It is also an 8 bit field.

Sign refers to the digital signature which is a proof of authentication. `F` refers to the Fragmentation bit which is a one bit field. If it is 0, then no fragmentation of data has occurred. If it is set to 1, fragmentation of data has occurred. FSN refers to the Fragment Sequence Number. This field will hold some number if the Fragment Bit is set to 1. Length refers to the length of data that is being carried by the Mobile Agent. The code corresponds to the Mobile Agent code which may be of some variable length. The data field represents the data that is being carried by the Mobile Agent.
4.5.1.6 Integrity Verifier

The integrity verification mechanism is employed to check the integrity of the incoming fragments into the buffer. The mechanism uses a checksum based on 1’s complement arithmetic to check the integrity of the received data.

4.5.1.7 Defragmentation

The Defragmentation module performs the reverse process of fragmentation. It checks the fragmentation bit. If the fragmentation bit is set to 0, it performs no action. If the bit is set to 1, it checks the Fragmentation Sequence Number, collects the data, sequence the data and integrate it for the system to process the data.

4.5.2 Working Mechanism

Consider the Secure Banking System which follows the proposed Secure Mobile Agent System Architecture. A client wishes to access some details from the bank. The client therefore generates a Mobile Agent and the user provides the data that needs to be carried. Now, the client registers the Agent ID, Host ID and the size of the Mobile Agent along with the data in the authentication Table. The Authentication Table acts as a lookup record which

<table>
<thead>
<tr>
<th>AID (8)</th>
<th>HID (8)</th>
<th>Sign(8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F(1)</td>
<td>FSN(7)</td>
<td>Length(16)</td>
</tr>
<tr>
<td>Code (variable)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data (variable) + Padding (optional)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verifier (Reserved)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.3 Mobile Agent Packet Format
could be referred by the Authentication Check in any platform which has registered to the Authentication Table. The size of the buffer in the passage should be pre-determined by the administrator before creating the Mobile Agent. The size of the Mobile Agent along with the data ‘a’ should be maximum equal to the size of the buffer ‘m’ in the passage. If a<=m the passage buffer will allow the Mobile Agent otherwise it will expel it. The external gate is closed as soon as the Mobile Agent enters the passage buffer. No other Mobile Agent is allowed to enter the passage buffer unless the verification inside the passage is completed. Fragmentation has to be done on the Mobile Agent to make it to the size of the buffer and the data are carried by dummy agents whose purpose is just carriage of data.

Fragmentation module is an essential part on the agent originator which divides the Mobile Agent into fragments. Each fragment must be of the size ‘n’ where n=m-(Sig id+ vi). If there are any remaining bytes available, zero padding will be done on the right side of the bytes for making them a whole fragment. A signature Sig id is generated for each fragment with the help of host id and a verifier vi is appended as shown in equation 4.1.

\[
\text{Fragment} = \text{Sig id} \parallel f_i \parallel v_i
\]  

(4.1)

The data is now being transmitted to the secure bank system. Now the Mobile Agent fragment reaches the external gate of the passage which performs verification of authentication using the digital signature of timestamp based Elliptic Curve Cryptography (explained in 4.5.3) it possesses. If the Mobile Agent does not pass the authentication check, it is not allowed inside the passage and killed immediately. If the Mobile Agent passes the check, it is allowed into the passage buffer. The Mobile Agent is kept in ‘zero execution’ state while this check is performed. To check the integrity of the incoming fragments inside the buffer the code verification mechanism (R’) is used.
Verifier \((R') = \text{Sig}_{id} + f_i + v_i \tag{4.2}\)

In equation 4.2, expression ‘+’ denotes one’s compliment addition. If \(R'\) consists of all zeros, the code received is non-malicious and will be added to a vector. Otherwise, the malicious fragment will be killed. The fragment collection process is repeated till the last fragment arrives to the vector.

The Defragmentation module checks whether the fragment bit is set or not. If the fragment bit is found to be set to 0, then the Mobile Agent is allowed to access the resource it needs after verification process succeeds. In case if the fragment bit is set to 1, then the Mobile Agent is kept in hold unless all the fragments in the sequence are received. Once all the fragments had arrived, the defragmentation module integrates the data and appends it to the actual Mobile Agent.

In the internal gate, the AID and size of the Mobile Agent is verified with the authentication table. The agent will be killed if any modifications found. It keeps all the dummy agents unless the Mobile Agent completes all the process in the platform. Once the task is accomplished and the result obtained, the Mobile Agent with the data is subjected to size check by the fragmentation module. If fragmentation is needed, it performs fragmentation and appends the data to the dummy agents which were kept waiting. In case, if there is no need for fragmentation the dummy agents will be destroyed. If the number of dummy agents is less than the number of fragment of data, new dummy Mobile Agents will be created by the secure Mobile Agent system’s Agent Generator. The resultant data is send to the client which requested the service.
The above mechanism, though pose some restriction over the size of the data, is found to be much efficient to combat with the Tailgating attacks. Since the Mobile Agent along with the data is of a consistent size, it is not possible for the malicious agent to travel behind or append with the legitimate Mobile Agent. Since, the dual check point system performs various checks; it is not so easy to by-pass or breach the security. Moreover, the zero execution environments help to secure the system from being collapsed by the malicious agent at the time of check.

4.5.3 **Digital Signature using Timestamp based Elliptic Curve**

Mobile Agent can construct the accessible network through the following steps:

The domain parameters for the initialization phase are as follows:

- **AO** - Agent Originator.
- **Private Key** - Na which should be less than n, where n is the order of a point G on the elliptic curve, the smallest positive integer.
- **E_p (a, b)** - elliptic curve, the parameters a, b and p the prime number.
- **G** - point on the elliptic curve whose order is large value n.
- **C_i and C_j** - Host and Mobile Agent
- **x and y** - values in the real numbers.
- **Public key** - Pa can be computed from Na and G.
4.5.3.1 Initialization phase

The Agent Originator assigns a private key to each Mobile Agent, and performs the following steps:

Step 1: Define elliptic curve $E_p(a, b)$ where $y^2 = x^3 + ax + b \pmod{p}$ in which, $a$ and $b$ must satisfy $4a^3 + 27b^2 \neq 0 \pmod{p}$, and $p$ is a large prime number.

Step 2: Choose a base point $G = (x, y)$ on $E_p(a, b)$, the point on the elliptic curve. The base point $G$ can be computed by satisfying the elliptic curve equation. The values when substituted become the perfect square and hence considered that the point lies on the elliptic curve.

Step 3: The calculations for the public key of $C_i$ and $C_j$ is as follows:

$$P_{a_i} = N_{a_i} \times G \pmod{p}$$

$$P_{a_j} = N_{a_j} \times G \pmod{p}$$

where $N_{a_i}$ and $N_{a_j}$ are the private keys.

$P_{a_i}$ and $P_{a_j}$ are the public keys.

and $G$ is the base point on the elliptic curve.

The domain parameters for the key assignment phase are as follows:

$u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_{10}, u_{11}, u_{12}$-random numbers specifying year, month, date, hours, minutes and seconds.

Y, M, D - represents year, month and date.

h, m, s - represents hour, minute and second.

“||” - is the concatenation operator.

K - Secret random number.

vt - calculates the date and time of expiry of the Mobile Agent.

ct - calculates the date and time of Mobile Agent creation.

et - calculates the date and time of Mobile Agent visiting the host.

M_w - denotes the date-time bound warrant.

R, S, Z - public parameters.

Na_j - private key of C_j

### 4.5.3.2 Key assignment phase

The Agent Originator performs the following steps for C_j:

**Step 1:** Choose twelve random numbers which are u1, u2, u3, u4, u5, u6, u7, u8, u9, u10, u11 and u12.

**Step 2:** Generate ct = v1||v2||v3||v4||v5||v6||v7||v8||v9||v10||v11||v12, where

v1 = H^Y(u1), v2 = H^{100-Y}(u2), v3 = H^M(u3), v4 = H^{12-M}(u4), v5 = H^D(u5), and v6 = H^{31-D}(u6), v7 = H^h(u7), v8 = H^{24-h}(u8), v9 = H^m(u9), v10 = H^{60-n}(u10), v11 = H^s(u11), v12 = H^{60-s}(u12), Y, M, D, h, m,
and \( s \) represents year, month, date, hours, minutes, seconds respectively, and "\( || \)" is the concatenation operator.

**Step 3:** Calculate \( vt = H^{100}(u1)||H^{100}(u2)||H^{12}(u3)||H^{12}(u4)||H^{31}(u5)||
H^{31}(u6)||H^{24}(u7)||H^{24}(u8)||H^{60}(u9)||H^{60}(u10)||H^{60}(u11)||H^{60}(u12). \)

**Step 4:** Agent Originator chooses another secret random number “\( k \)”, and uses the previously calculated parameters \( t \) and \( v \) to calculate signature date-time bound warrant \( M_w = (vt, ct) \), and also the following public parameters, \( R, S, Z \), and the private key \( Na_j \) of Mobile Agent \( C_j \). The calculations are as follows:

\[
Z = k\times G = (x_1, y_1) \text{ where } x_1 \text{ and } y_1 \text{ are the points that lie on the elliptic curve.}
\]

\[
R = x_1 \mod p
\]

\[
S = k^{-1} \times (H(M_w) - (Na_i \times R) \mod p), \text{ where } Na_i \text{ is the private key of } C_i.
\]

\[
Na_j = H(k, ID_j)
\]

Where \( ID_j \) is the public unique code of Mobile Agent identity \( C_j \).

The domain parameters for the key derivation phase are as follows:

\( C_i \) and \( C_j \) – The Host and Mobile Agent who take part in the key derivation.

\( R, S, M_w \) – public parameters to calculate secret key \( k \).

\( Na_j \) – private key of \( C_j \).

\( Na_i \) – private key of \( C_i \).
4.5.3.3 Key derivation phase

Step 1: Use public parameter R, S and $M_w$ to calculate secret parameter $k$, as follows:

$$k = ((S + N_{ai} \times H(M_w)) \times R \mod p)^{-1},$$

$N_{ai}$ is the private key of $C_i$.

Step 2: Calculate $C_j$’s private key $N_{aj} = H(k, ID_j)$.

The domain parameters for the key expiration validation phase are as follows:

- $e_t$: the parameter that is used to check the expiration of key.

4.5.3.4 Key expiration check phase

Step 1: Calculate $e_t = H^{100-Y}(u1) || H^{Y}(u2) || H^{12-M}(u3) || H^{M}(u4) ||$

$$H^{31-D}(u5) || H^{D}(u6) || H^{24-h}(u7) || H^{h}(u8) || H^{60-m}(u9) || H^{m}(u10) ||$$

$$H^{60-s}(u11) || H^{s}(u12).$$

Step 2: If $e_t$ is less than or equal to $v_t$, then the key is valid and the Mobile Agent is permitted. Otherwise, the key is said to be expired.

The domain parameters for the key signature verification phase are as follows:

- $Z, R, S$ - public parameters.

4.5.3.5 Key signature check phase

By using the following equation

$$Z = k \times G = (x_1, y_1)$$

$$R = x_1 \mod p$$

$$S = k^{-1} \times (H(M_w) - (N_{ai} \times R) \mod p)$$
Calculate $V_1 = R \times P_{a_i} + S \times Z$ and $V_2 = H(M_w) \times G$.

Judge whether $V_1$ is equal to $V_2$.

If the two are equal, then the signature is true.

**Proof:**

$$V_1 = R \times P_{a_i} + S \times Z$$

$$= R \times P_{a_i} + [k^{-1} \times (H(M_w) - (N_{a_i} \times R) \mod p)] \times Z$$

\{Since $S = k^{-1} \times (H(M_w) - (N_{a_i} \times R) \mod p)$\}

$$= R \times P_{a_i} + [k^{-1} \times (H(M_w) - (N_{a_i} \times R) \mod p)] \times (k \times G)$$

\{Since $Z = k \times G$\}

$$= R \times P_{a_i} + [k^{-1} \times (G \times H(M_w) - (N_{a_i} \times R) \times G \mod p))] \times k$$

$$= R \times P_{a_i} + [k^{-1} \times G \times H(M_w) - P_{a_i} \times R \times k^{-1}] \times k$$

\{Since $P_{a_i} = N_{a_i} \times G \mod p$\}

$$= R \times P_{a_i} + G \times H(M_w) - (P_{a_i} \times R)$$

$$= G \times H(M_w)$$

$$= V_2.$$  

$V_1$ is equal to $V_2$. Thus, the signature is true.

### 4.5.4 Materials and Methods

An experimentation to compare the ordinary authentication verification based on digital signature and our proposed architecture was done. The banking system was taken for this task. A Mobile Agent system was designed using the IBM Aglets 2.0.1. As described in the scenario, the
client uses the Account number, Personal Index Number and a digital signature to authenticate itself with the Banking System. In the secure system, the passage buffer was set to 16KB. Therefore, the fragmentation module fragments the data and Mobile Agent to 16KB combined. The experimentation was done parallel in 2 rows, with 12 PCs with Intel Dual Core Processors in each row using Windows XP Operating System. Totally, 24 systems were used. 11 systems acts as clients and one system contains the central banking system in each row. One row contains the banking system with our proposed architecture and the other with the traditional digital signature based authentication system.

4.5.5 Results and Discussion

The experimentation is performed for a period of nearly six hours. Malicious Mobile Agents which have the capability to inject or append code to other Mobile Agents were introduced inside the network from an external source. Various results are obtained and performance comparisons based on various parameters are done.

![Figure 4.4 Detection of Malicious Agent](image)
The first comparison is based on the number of malicious agents introduced against the number of Malicious Agents detected by the system. Figure 4.4 clearly shows the distinction in detection between the proposed Dual Check-Point Analysis system (DCPA) and a traditional system. The Traditional System which is used in the experimentation is composed of a system based on the Agent Identity check. This does not include any Secure Channel Authentication Table like that used in Dual Check Point Analysis. The Agents use a pass code to enter the computational platform in this system.

The graph is based on the following experimentation. Malicious Agents were introduced to move around the system in order of increasing numbers. At first, one malicious agent was introduced in the system and was verified whether any change is observed. It is observed that the system with the proposed architecture detects the malicious agent and kills it. The same malicious agent when introduced in the traditional system could not identify the malicious agent and had fallen prey to the malicious agent. Now, both the systems were reset to their initial conditions. Two malicious Mobile Agents were introduced. At this moment also, the traditional system was found to be non-functional with respect to detection. This experiment was done again and again with increasing count of malicious Mobile Agents upto 40. It was observed that the performance of the proposed system was consistent in detection when compared to the traditional system. It could be seen that the traditional system could not detect any malicious Mobile Agents up to 14. It was because those malicious Mobile Agents were specially designed to perform tailgating. Whereas after 14, it is found that the traditional system could detect those malicious Mobile Agents which possessed a false digital signature or those which provided irrelevant Account Number or Personal Index Number.
The response to the Malicious Agents introduced inside the system is shown in Table 4.1. The above said experimental results could be understood by the table given below. It should be noted that the Malicious Agents are capable of performing Tailgating Attacks. The traditional system may be doing well with most type of attacks, but it does not concentrate much over the tailgating attack since the attack is not known in the past.

**Table 4.1 Detection of Malicious Agents (Traditional System vs Dual Check Point Analysis)**

<table>
<thead>
<tr>
<th>No. of Malicious Agents Introduced</th>
<th>No. of Malicious Agents detected</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DCPA</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
</tr>
</tbody>
</table>
In the Table 4.2 the computational complexity of DCPA is described as the time taken to compute the code fragmentation, digital signature for authentication, append integrity verifier and authentication registration while agent creation in the home platform. The execution platform has the time to verifying the digital signature, checking of signature
by the host, check the integrity verifier and finally the Authentication check is performed. The time complexity for both home and remote platform is $O(1)$.

### Table 4.2 Computational complexity of Dual Check Point Analysis

<table>
<thead>
<tr>
<th>Place of Agent execution</th>
<th>Time</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home platform</td>
<td>$t_{CF} + t_{DS} + t_{AIV} + t_{RA}$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Execution platform (Remote)</td>
<td>$t_{VDS} + t_{SV} + t_{CIV} + t_{VAC}$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

4.5.6 Security Analysis

A security analysis is performed to examine whether the proposed scheme is secure or not for the practical applications. The analysis focuses upon three types of attack that may impact the system security.

1. **Reverse attack**

   The Mobile Agent $C_j$ could try to use its public key $P_{aj}$ to derive $C_i$’s public key $P_{ai}$ and they attempt to steal the data that is accessible only by $C_i$. $C_j$ does not have the right to access the data that is only available to $C_i$, because it is very difficult to obtain public key $P_{ai}$ of $C_i$. Hence, $C_j$’s reverse attack would fail and therefore, the other agent/host can never derive the public key of the agent/host.

2. **Date-Time alteration attack**

   Suppose a class $C_i$ attempts to extend the validity of the key or continues to use the original encryption of the key that is in the date-time bound warrant

   $$M_w = (vt, ct)$$
In our proposed method,

\[
\Rightarrow H^{100-Y}(u1) || H^Y(u2) || H^{12-M}(u3) || H^M(u4) || H^{31-D}(u5) || H^D(u6) || \\
H^{24-h}(u7) || H^h(u8) || \\
H^{60-m}(u9) || H^m(u10) || H^{60-y}(u11) || H^y(u12) \rightarrow et
\]

\[
\Rightarrow H^{100}(u1) || H^{100}(u2) || H^{12}(u3) || H^{12}(u4) || H^{31}(u5) || H^{31}(u6) || H^{24}(u7) || \\
H^{24}(u8) || H^{60}(u9) || H^{60}(u10) || H^{60}(u11) || H^{60}(u12) \rightarrow vt
\]

Where \( v1 = H^Y(u1) \), \( v2 = H^{100-Y}(u2) \), \( v3 = H^M(u3) \), \( v4 = H^{12-M}(u4) \), \( v5 = H^D(u5) \), \( v6 = H^{31-D}(u6) \), \( v7 = H^h(u7) \), \( v8 = H^{24-h}(u8) \), \( v9 = H^m(u9) \), \( v10 = H^{60-s}(u10) \), \( v11 = H^i(u11) \), \( v12 = H^{60-s}(u12) \).

The user tries to extend the key or continues to use the original encryption key and alters the system parameter \( ct \) and public parameter \( v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12 \).

The expiry time of the key is originally defined by the Agent home platform, and it is protected by the twelve secret random numbers, \( u1, u2, u3, u4, u5, u6, u7, u8, u9, u10, u11, u12 \), and one-way hash function.

Suppose a class \( C_i \) tries to extend the expiry year from \( y \) to \( y' \), and \( C_i \) can calculate \( v1' = H^Y(u2) \) from \( v1 = H^Y(u1) \). However, \( C_i \) cannot calculate \( H^{100-Y}(u2) \). Therefore, no entity shall be able to alter the value of \( vt \), and the date-time bound warrant \( M_w = (vt, ct) \) cannot be changed and this type of attack can be avoided.

(3) **Code Integrity**

When a Mobile Agent travels from Agent originator to the remote host, any alteration may happen to the Mobile Agent’s code/data due to
malicious agent. The equation 4.3 shows the alterations happened to the Mobile Agent can be detected by the verifier.

\[ R' = \text{Sig}_{id} + f_i + v_i \]  

(4.3)

4.6 IMPROVED MALICIOUS IDENTIFICATION USING TRIPMARKER

Mobile Agent technology is one of the most advanced technologies which introduce the concept of code mobility. The mobility of the code facilitates various tasks to be performed in an effective manner. Mobile Agents prove to be more efficient in the case of usage in a distributed environment where data from various systems may be needed to perform a particular task. This also poses a threat over the entire system that the mobile code may be prone to malicious attacks. The malicious attacks may either manipulate the data or increase the network traffic or misguide the Mobile Agent to some other location. Hence, Mobile Agent security plays an important role in the Mobile Agent system. The security may be provided either to the Mobile Agent or agent platform. In our work, Mobile Agent security protocol is proposed which provides security to the Mobile Agent by protecting it from various types of attacks. A combination of techniques has been used to make this protocol an efficient one. We use trip marking to overcome these replay attacks, digital signature to provide the service of authenticity and authorization and finally the Malicious Identification police, to check whether any malicious attacks have been done over the Mobile Agent. The theoretical implications show that the Mobile Agent is theoretically secure. From the experimental point of view, it has been found that the protocol provides security to the Mobile Agent which makes it resistant to the most of the known types of attacks.
4.6.1 Existing systems and Issues

There are numerous works that have been proposed to provide Mobile Agent with security. In this part, a few works are being analyzed and the inference made from them is scripted.

Malicious Identification Police (Venkatesan et al 2010) which helps to identify the malicious agents by means of the extended Root canal Algorithm (XRC). The system is based on the policy files or the definition files that are present within the system. This system overcomes the drawback of the simple MIP which works based on the principle of signature based Intrusion Detection System. Moreover, the system uses an Attack Identification Scanner which would help the system to check whether any attack have been made over the Mobile Agent or the Mobile Agent is subjected to any sort of attack. This system also compromises of techniques to identify any new types of attack by means of lexical analysis and similar methods. However, the attack code detection strategy is not well defined. Though, the system maintains the integrity of the Mobile Agent, it couldn’t provide a better solution against replay attacks and non-repudiation. This system serves to act similar to an anti-virus that is being used in a system.

Hwang et al (2003) provided a practical threshold proxy signature scheme with known signers; they claim that this scheme satisfies all the five proxy requirements. Our scheme satisfies all proxy requirements and uses only simple Lagrange formula to share the proxy signature key. Unfortunately, Wang et al (2004) commented that Hwang et al scheme had some drawback in secrecy, Proxy Protection, Non-reputation, Known Signer and Unforgeability that make it insecure.
Kuo and Chen (2005), pointed out the following two security weaknesses in Hwang et al scheme. One weakness was secure application problem of RSA, and the other weakness was that the value of the Lagrange coefficient was not clearly defined.

Chang and Chang (2007), An RSA-based (t, n) threshold proxy signature scheme with freewill identities, this scheme could overcome the security flaw. Secrecy: the original signer’s private key cannot be derived. Proxy Protection: no one can generate the valid partial proxy signature except the designated proxy signer. Unforgeability: only if t or more proxy signers cooperate to generate a valid proxy signature. Non-repudiation: proxy signers cannot deny signing the message, nor can the original signer deny delegating the capacity to the proxy signers. Time Constraint: the proxy signing keys can only be used during the delegation period. Known Signers: the actual signers of a given threshold proxy signature can be determined for an internal audit. But the function of the proxy signature combiner is not clearly defined.

The proxy signature protocol (Xuan Hong 2009) mainly focuses on the non-repudiation attack. This protocol satisfies the various aspects like verifiability, Unforgeability, secrecy, undeniability and identifiability which are mainly concerned with non-repudiation attacks. This method uses a digital signature to assure authenticity. Though this method is more successful with non-repudiation attacks, it couldn’t prove itself to be efficient in case of other types of attacks. Therefore, this technique couldn’t be considered so efficient for a secure transaction of data to be done. This technique doesn’t assure most other secure services and the issues related to it. However, this technique is provably efficient in case of identifying a host or the origin of a Mobile Agent.
The Self-Protected Mobile Agents System (Ametller, Robles and Ortgea Ruiz 2004) is one in which the agent code is protected by means of a public decrypting function, agent authentication and some other classical protection mechanism. This method of protecting the Mobile Agent may be efficient against classical threats. Attacks to overcome the techniques used in this system have been identified in the recent period and hence, this method couldn’t be used in case of the modern Mobile Agent systems.

4.6.2 Proposed Solution

The various components of the system are as follows.

- Agent Platform
- Trip Marker
- Proxy Signer
- Malicious Identification Police

4.6.2.1 Agent Platform

The agent platform is one which is responsible for the creation of the Mobile Agent. The Mobile Agent that is created is subjected to the various processes in the system. The secure Mobile Agent is sent to various parts of the system for further processing. Figure 4.5 shows the architecture of the system.
4.6.2 Tripmarker

The trip marker appends the trip mark data to the Mobile Agent. The trip marker sets an expiry time and a counter for the Mobile Agent. It is also responsible for checking and resetting the expiry time of the Mobile Agent as well as decrementing the counter for the Mobile Agent. This would be helpful to overcome the external replay attacks.
4.6.2.3  **Proxy Signer**

The proxy signer is responsible for signing the Mobile Agent as well the data. This is used to assure the authentication service of the system as a whole. The proxy signer uses a sequence of processes to perform the signing process more secure.

4.6.2.4  **Malicious Identification Police**

The Malicious Identification Police contains an inbuilt Attack Identifier which helps to detect any sort of attack that has been made over the Mobile Agent during the time of transportation from one host to another. Moreover, it uses various policies to check whether a Mobile Agent is authorized to access a resource or not. The policy files are maintained in a database. This also decides whether a Mobile Agent needs to be transmitted from the system. It encrypts and transmits the Mobile Agent along with the data.

4.6.3  **Agent Format**

The figure 4.6 shows the format of the Mobile Agent that is being used in this protocol. The Mobile Agent format contains the following fields:

- Agent Identifier (AID)
- State
- Digital Signature
- Expiry
- Counter
- Authorization Node
- Agent code
- Data
4.6.3.1 Agent Identifier

The agent identifier is the one which uniquely distinguishes the Mobile Agent. It is an 8 bit field which is being occupied by a unique number with respect to the system.

4.6.3.2 State

The state field represents the state of the agent. The agent may be active or passive or may get dispatched or retracted. This field could be directly accessed by the agent platform. It is also an 8 bit field.

4.6.3.3 Digital Signature

The digital signature is appended to the Mobile Agent by means of the proxy signer. The proxy signer may use any means of signing. This part ensures the authentication services. The size of this field is 16 bits.

4.6.3.4 Expiry Timer

The expiry time is set by the trip marker which sets a time stamp which contains the maximum time limit for which the Mobile Agent must be alive. It is an 8 bit field.
4.6.3.5 Counter

The counter contains the number of itineraries the Mobile Agent could make to its maximum. Once this field becomes zero, the Mobile Agent reaches its destination or gets destroyed. The counter is also an 8 bit field.

4.6.3.6 Authorization Node

The authorization node is the creation of the Malicious Identification Police. The node contains details of the privileges of the Mobile Agent which indicates to which resources the Mobile Agent could have access. This is a 16 bit field.

4.6.3.7 Agent Code

Agent code is the actual code of the Mobile Agent is made of. It is the source code which is programmed to move from one host to another. The size of this field is variable and depends on the design of the system.

4.6.3.8 Agent Data

The data field is also a variable one which would contain the data the agent needs to carry in order to perform a task.

4.6.4 Proxy Signing

There are various methods to perform digital signing over the data. One such technique is the proxy signing technique which involves a sequence of steps.

Consider the original signer id $I_A$ and the proxy signers or the platforms involved in signing $I_{p1}, I_{p2}, \ldots, I_{pn}$, a combiner C, secure one way hash function $H(.)$ and the message warrant $m_w$ which is used to record $I_A$.  

Two random prime numbers \( p_0, q_0 \) of equal length are chosen by \( I_A \).

\[
p_0 = 2p_0' + 1 \\
q_0 = 2q_0' + 1
\]

where \( p_0', q_0' \) are prime numbers.

Let, \( N_0 = p_0q_0 \) and \( M_0 = p_0^e q_0^e \)

where \( M_0 \) is of the order of \( Q_{N_0} \). \( Q_n \) is the subgroup of squares in \( \mathbb{Z}_n \) which is in order

\[
M = [(p-1)(q-1)]/4
\]

We use the RSA algorithm and so the agent with identity \( I_A \) computes the RSA exponents \( e_0 \) and \( d_0 \) where,

\[
e_0 d_0 \equiv 1 \mod M_0
\]

Now, the private key is \( (d_0, M_0) \) and the public key is \( (N_0, e_0) \). In general,

\[
N_i = p_i q_i \\
\phi(N_i) = (p_i - 1)(q_i - 1) \\
e_i d_i \equiv 1 \mod \phi(N_i)
\]

and \( (N_i, d_i) \) becomes the private key, \( (N_i, e_i) \) becomes the public key. Consider the threshold proxy key of \( I_A \) as

\[
D = d_0 H(m_n) \mod M_0
\]

\( I_A \) shares the signing key \( D \) among \( n \) proxy signers.

\( I_A \) sets \( a_{Ao} = D \) and chooses \( a_{Ai} \) at random from
\{0,1,\ldots, M_0 - 1\} \text{ for } 1 \leq i < t

which defines $t - 1$ degree polynomials.

$$f(x) = a_{0} + a_{1}x + \cdots + a_{t-1}x^{t-1} \mod M_0$$

The partial proxy signature key $K_i = f(i) \mod N_0$ for each proxy signer $I_{pi}$ and is computed by $I_A$.

The validation is performed by computing the proof.

$I_A$ chooses randomly $v \in \mathbb{Q}_{N_0}$ for $1 \leq i \leq n$

$$v_i = v^{k_i} \in \mathbb{Q}_{N_0}$$

$I_A$ makes $(v_1, v_2, \ldots, v_i)$ public. Coming to the part of proxy signature generation $x = H(m, m_w)$ and $\Delta = n!$

Each proxy signer uses its partial proxy signing key $K_i$ to sign the partial signature.

$$x_i = x^{\Delta/s_i} \in \mathbb{Q}_{N_0}$$

$I_{pi}$ computes the proxy signature $(\Delta \sigma_i, \sigma_i)$

$$\Delta \sigma_i = [x_i, N_j], \sigma_i = x_i^{d_i} \mod N_i$$

The proof of correctness could be done as follows

$$\log_{v_i} x_{i}^{2} = \log_{v} v_i$$

chooses $r \in \{0,1,\ldots, 2^{|N_0| / 2L_t} - 1\}$ where $L_t$ is the secondary security parameter. The function then computes
\( v' = v' \)
\( x^* = x^* \)
\( c = H'(v, \bar{x}, v^{'}, x^2, v', x^{'}, z) = k_i C + r \)

The proof of correctness is given as \((z, c)\) and the final partial proxy signature is given as \((i, \Delta \sigma_i, \sigma_i, z, c)\).

While considering the final part of the signature, we move on to combining. \(C\) is the proxy signature combines which is a signer who do not own any secret parameters. As soon as the combiner receives \((i, \Delta \sigma_i, \sigma_i, c, z)\) it recovers the partial signature by

\[ x_i = \Delta \sigma_i \times N_i + (\sigma_i^0 \mod N_i) \]

To calculate the proof of correctness, the following is done

\[ C = H'(v, \bar{x}, v^{'}, x^2, v^{'}, x^{'}, \bar{x}^2, x_i^{2^c}) \]

Suppose, all the proxy signatures are correct and valid, then the corresponding signer set is

\[ S = \{i, i_2, ..., i_n\} \subset \{1, 2, ..., n\} \]
and the signature share is combined. We define

\[ \lambda^S_{ij} = \frac{\Delta \pi_j \in S \setminus \{j\} (i - j')}{\Delta \pi_j \in S \setminus \{j\} (j - j')} \in Z \]

Proxy signature \( w \) of the message which is confirmed to \( m_w \) is given as

\[ w = x_{i1}^{2^i \cdot j_1} \cdots x_{in}^{2^i \cdot j_n} \mod N_0 \]

Using the standard Lagrangian formula
\[ \Delta f(i) = \sum_{j \in \delta} I_{i,j}^S \cdot f(j) \mod M_0 \]

since \( x_{ij}^2 = x_{ij}^{2 \cdot \lambda_{ij}} \)

we have \( w_{e_0} = x^{4 \cdot \lambda_{ij} \cdot H(m_u) \mod N_0} \)

since \( \gcd(4 \Lambda^2, e_0) = 1 \)

\[ y_{e_0} = x^{H(m_u)} \]

(i.e. \( y_{e_0} = H(m, m_u)^{H(m_u)} \))

using a standard algorithm

\[ y = w^a x^b \]

where \( a \) and \( b \) are integers such that \( 4 \Lambda^2 a + e_0 b = 1 \)

which is obtained from the extended Euclidean algorithm on \( 4 \Lambda^2 \) and \( e_0 \). The proxy signing involves the steps above for the digital signing process to be successful.

### 4.6.5 Process

Initially the Mobile Agent is generated by the agent platform and the data it needs to carry is also allocated by the agent platform. The Mobile Agent is subjected to encapsulation when it gets off a system and is decapsulated when it enters the system. The encapsulation and decapsulation are defined below. The host_agent represents the agent originating from the host and remote_agent represents the agent originating from a remote host.
4.6.5.1 Encapsulation

READ host_agent

APPEND expiry_time AND loop_counter

APPEND digital_signature

APPEND authorization_node

DECIDE transmit (host_data)

ENCRYPT AND TRANSMIT (host_data)

4.6.5.2 Decapsulation

READ remote_agent

DECRYPT remote_agent

VERIFY authorization_node

SCAN remote_agent

IF malicious

DESTROY remote_agent

ELSE

VERIFY digital_signature

IF valid

VERIFY expiry_timer AND loop_counter

DECREMENT loop_counter

ELSE

DESTROY (remote_agent)

ENDIF

ENDIF
Once the Mobile Agent along with the data is initialized, the trip marker appends the expiry time and counter. Along with that, the proxy signer appends the digital signature over the Mobile Agent, at the time of creation contains an identity and a state. When the Mobile Agent along with the trip marking data and digital signature is about to leave the system, the Malicious Identification Police appends the authorization details inorder to provide privileges to the Mobile Agent to access the various resources in the foreign platform. Once all these processes are complete, the Mobile Agent is allowed to be transmitted through the network.

Similarly, when a Mobile Agent enters the system, the Attack Identifier verifies it whether the Mobile Agent have been subjected to any sort of attack. When the Mobile Agent is not attacked by any malicious objects, it is allowed to access the resources inside the agent platform as per the details inside the authorization node. Further the digital signature is being checked to verify whether the Mobile Agent is an authenticated one. The trip marker checks the expiry times for the time stamp value and the counter values. The counter value is decremented for each itinerary for the Mobile Agent. Each time the Mobile Agent wishes to access a resource the authorization node is checked to verify whether the Mobile Agent has the privilege to access the particular resource.

### 4.6.6 Experimental Results

Two types of Mobile Agent systems are designed using Aglets 2.0.1 in a Network containing 20 systems with Intel Dual Core processor functioning in Windows XP operating system. The systems are separated into two groups each containing 10 systems. The Mobile Agents are made to transfer from one system to another to perform some operations. The Mobile Agents carry data with them. Malicious agents which would attack the Mobile Agents in various methods are introduced inside the network.
The results obtained for implementation are shown below.

**Figure 4.7 Setting of Access Rights**

Figure 4.7 shows the sequence of operations that are performed over the Mobile Agent before it is being dispatched to the foreign platform. The tripmarker is set to 4, so that the Mobile agent can have a maximum of 4 itineraries. Further, the digital signature is appended to the Mobile Agent, which is essential for maintaining the integrity of the data. After this process, the authorization node settings are being applied which provides access rights to the Mobile Agent. The access rights are set by the Mobile Agent platform and are verified by the privilege verifier.

**Figure 4.8 Encrypted Private Key**
The private key generated by the Agreement Module is shown in Figure 4.8. The private key is of size 128 bytes and is encrypted to improve the security of the key.

![Figure 4.9 Digital Signature of Data](image)

Figure 4.9 Digital Signature of Data

Figure 4.9 shows the Digital Signature created as a result of the Proxy Signing process. This is the digital signature generated for the message that is carried by the Mobile Agent. The Digital Signature is generated by a combination of SHA1 and DSA. This is appended to the data carried by the Mobile Agent. This could be verified to check the integrity of the message.

![Figure 4.10 Dispatching of Demo Agent](image)

Figure 4.10 Dispatching of Demo Agent
Figure 4.10 shows the dispatching of the Demo Agent from the home platform to some other platforms that are specified in the itinerary which is set by the Itinerary Setter.

Figure 4.11 Message Handling by Worker Agent

Figure 4.11 shows the activities performed in the foreign agent platform, once the Mobile Agent reaches it. In the foreign agent platform, initially the Malicious Identification check is performed as soon as it enters the platform. The integrity check is performed to find whether any modification is made in the data carried by the Mobile Agent. After checking the integrity, the tripmarker is verified and the itinerary is decremented by 1. This helps to overcome the external replay attacks. Finally, the inference that the agent is trustable is made and the Mobile Agent is allowed to access the resources in the agent platform.
Figure 4.12 Collection of results at home platform

Figure 4.12 shows the collection of results which it had obtained during the whole phase of itinerary. Since it has visited four Mobile Agent platforms during the itinerary, it could be observed that there are four “Result Message...” in the home platform. After completing the collection process, the Demo Agent gets terminated.

In Figure 4.13 EMASP refers to the Enhanced Mobile Agent Security Protocol, PSP refers to the Proxy Signature Protocol and SPMA refers to the Self Protected Mobile Agents. A comparison between the three schemes is shown in the figure.

Initially, the Malicious Mobile Agents are made to roam through the network. The Malicious Agents are capable of attacking any type of Mobile Agent and cause either tampering of code or code modification or tend the agent to undergo replay attacks. A system to send a message and to collect the resultant message is taken for the experimentation setup. The EMASP system is first taken into consideration. The Mobile Agents are made to move from one system to another system. Initially only one Malicious Agent is made to
travel along the network. Later this count is incremented upto twenty five by letting the malicious agents one by one.

![Graph showing Malicious Agents Vs Mobile Agent Failure](image)

**Figure 4.13 Malicious Agents Vs Mobile Agent Failure**

It is found that, out of the twenty five Mobile Agents roaming inside the network none of the Mobile Agents are attacked by the Malicious Agent that is introduced initially. Later, when the number of Malicious Agents increased to twenty two, one Mobile Agent had fallen prey to the attack. There is considerably a minimal chance for this to happen. On analysis, it is found that the Mobile Agent is subjected to some unidentified attack which is not addressed in the former research works on Mobile Agent Security. However, the attack is found to be tailgating and the solution is provided in Chapter 4.5. On integrating the architecture with this protocol, tailgating could also be addressed.

Consider the case of Proxy Signature Protocol. Initially the graph is at the lower scale showing that there is no attack over the Mobile Agent. But gradually, it is found to be increasing. This is because; initially Malicious Agents which cause attacks compromising integrity and authentication are introduced. But later on, Malicious Agents which cause replay attacks are introduced. This caused failure to the Proxy Signature Protocol System.
third case is the Self-protected Scheme. Since, no definite protective mechanism to withstand the new types of threats are used in this scheme, this scheme also resulted in failure with the introduction of Malicious Agents. The overall results obtained over Malicious Agents vs Mobile Agent Failures are tabulated in Table 4.3.

**Table 4.3 Comparison of Enhanced Mobile Agent Security Protocol, Proxy Signature Protocol and Self Protected Mobile Agents**

<table>
<thead>
<tr>
<th>No. of Malicious Agents Introduced</th>
<th>EMASP</th>
<th>PSP</th>
<th>SPMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>2</td>
<td>17</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>4</td>
<td>19</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>9</td>
<td>23</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>12</td>
<td>24</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>13</td>
<td>25</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>14</td>
<td>26</td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>15</td>
<td>28</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>16</td>
<td>29</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>17</td>
<td>31</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>18</td>
<td>33</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>19</td>
<td>35</td>
</tr>
</tbody>
</table>
From the above experimentation, the Attack Immunity percentage is calculated. Attack Immunity Percentage is the percentage ratio between the number of Mobile Agents immune to attack and total number of Mobile Agents.

![Graph showing comparison of Attack Immunity]

**Figure 4.14 Comparison of Attack Immunity**

From Figure 4.14, it could be observed that the performance rating in terms of attack Immunity for EMASP is greater when compared to PSP and SPMA. During experimentation 25 malicious agents are introduced inside the system, which comprises of 36 mobile agents. On introduction of the malicious agents it is found that some of the mobile agents inside the system are attacked. At completion, it is found that the system using SPMA secures only one mobile agent, the system using PSP secures 17 mobile agents, whereas the system using EMASP makes 35 mobile agents immune to the attacks by malicious agents. There is only a minimal chance for not capable of immune. However, this may be a special case of tailgating attack. It is seen that EMASP is more secure than the other two schemes.
4.6.7 Analysis

4.6.7.1 Authentication

The system is made to function using the schemes shown above. With the introduction of different malicious agents, it is found that the Enhanced Mobile Agent Security Protocol is immune to most types of attacks compared to the Proxy Signature Protocol and the Self Protected Mobile Agent Scheme.

4.6.7.2 Access Control

Access control is another security service which needs to be concentrated in any security mechanism. Any system designed must provide the access control service. The authorization node present in our protocol is the main tool to provide this service. It contains all the privilege definition which controls the access towards a resource.

4.6.7.3 Confidentiality

The security protocol provides an encryption scheme by which the data is encrypted. Encryption scheme protects the data from being viewed by other users which ensures confidentiality of the data.

4.6.7.4 Integrity

The protocol also provides a sophisticated integrity measure by not allowing any other malicious agents to access to the data by means of the encryption technique which it uses before transmittance of the data.

4.6.7.5 Non-repudiation

The protocol provides the non-repudiation service by means of the digital signature. The digital signature ensures the origin of the Mobile Agent,
thereby avoids any sort of attack which poses repudiation towards the agent or the data.

4.6.7.6 Masquerading

The protocol is efficiently designed to overcome the masquerading attack. Each agent is provided with a unique agent identifier which distinguishes itself from other Mobile Agents in the system. Therefore no other Mobile Agent in the system could have the same identifier, thereby making the system immune to Masquerading.

4.6.7.7 Replay Attacks

The protocol is designed especially in concentration with replay attacks. The tripmarker used in the system facilitates the protocol to encounter replay attacks. The value in the counter and the expiry time field decides the further itineration of the Mobile Agent. In case, if it is found to be invalid, then the Mobile Agent is destroyed or marked to be malicious.

4.7 SUMMARY

In this chapter, an efficient architecture to overcome the tailgating attack has been designed for a Mobile Agent system. The introduction of the Dual Check-point system and the application of constant size for Mobile Agents are proved to be effective in encountering the tailgating attacks. Also, the fragmentation and defragmentation process included in this system supports the architecture to the maximum extent and thereby help the system to identify the tailgating attacks. Another main objective of this chapter, which is to bring awareness about tailgating attacks in a Mobile Agent environment, has also been accomplished by description of the nature of attack in the Mobile Agent system and an efficient method to overcome this attack has also been achieved in this work.
A protocol has been developed which is immune to most types of known attacks. It uses the techniques of trip marking, digital signing and Malicious Identification Police to overcome the most types of attacks. From the experimental point of view, it has been found that the Mobile Agents in this system are more immune to attacks while compared to the systems which are implemented in an ordinary environment.