A Hybrid Security Model for Mobile Agents in Distributed Systems

Ini J. Umoeka[1], Imo J. Eyoh[2]

[1,2]Department of Computer Science
University of Uyo, Uyo, Nigeria
iniumoeka@uniuyo.edu.ng
imoheyoh@uniuyo.edu.ng

Abstract: ***This paper focuses on the development of a hybrid security model for Mobile Agents (MAs) system in a distributed network environment using RSA and DSA. The MA computing vulnerabilities include authentication, secure messaging, certification, trusted third parties, non-repudiation and resource control. These vulnerabilities have prevented widespread adoption of mobile agent technology. The proposed framework is able to counter these threats as agents must be protected at the same time. The model is an attempt at overcoming these issues for efficient messaging and host security. The methodology adopted is an Object-Oriented System Analysis and Design (OOSAD). The new system is implemented using JAVA programming technologies and execute on any environment with appropriate JAVA Virtual Machine.***

Keywords: mobile agents, encryption, decryption, authentication.

1. Introduction

Over the years, computer systems have evolved from centralized monolithic computing devices supporting static applications, into client-server environments that allow complex forms of distributed computing. Throughout this evolution limited forms of code mobility have existed: the earliest being remote job entry terminals used to submit programs to a central computer and the latest being Java applets downloaded from web servers into web browsers. A new phase of evolution is now under way that goes one step further, allowing complete mobility of cooperating applications among supporting platforms to form a large-scale, loosely-coupled distributed system.

The catalysts for this evolutionary path are mobile software agents. An agent is a software object on an execution environment that is acting on behalf of a person or an organization (Jennings and Wooldridge (1998) Braun and Rossak, 2005). The agent could be reactive (it is sensible to the environment and acts depending on its changes); autonomous (it has its own thread of execution and therefore it can have initiative to execute tasks and control its own action); proactive (its behaviour is object oriented) and continuous in time (it is constantly under execution). Furthermore, an agent can be communicative if it can interchange messages with other parties, learning if it is able to adapt using previous experience; stationary, always resident at a single platform or mobile if it can move (migrate) from one execution environment to another (White, 1995). Software agents are programs that are goal-directed and capable of suspending their execution on one platform and moving to another platform where they resume execution.

In section 2, we discuss some of the security issues and highlight mobile agents security threats. In section 3 and 4, we propose and design a security framework for Mobile Agent Software (MAS) in a distributed computer network. In section 5, we provide a conceptual view of the four major processes involved in the encryption and decryption respectively. Model implementation is described in chapter 6 with Summary, conclusion and future works in chapter 7.

2. Overview of the Security Issues for Mobile Agents

Although the mobile agent paradigm extends the capabilities of traditional ways of remote communication and distributed computing, it also raises new security issues (Chess, 1998). These are generally divided into two broad areas:

i). protecting the host from malicious agents, and ii). protecting the agent from hostile hosts. Protecting the host from attacks by malicious agents is possible by using effective access control and sandbox mechanisms (e.g. Java's sandbox security component). The mobile agent computing paradigm raises several privacy and security concerns, which are the main obstacles to the widespread use and adoption of this new technology. A more challenging problem is how to protect an agent from being abused by a hostile server.

Threats to security generally fall into three main classes: disclosure of information, denial of service, and corruption of information. There are a variety of ways to examine these classes of threats in greater detail as they apply to agent systems. Here, we use the components of an agent system to categorize the threats as a way to identify the possible source and target of an attack. It is important to note that many of the threats that are discussed have counterparts in conventional client-server systems and have always existed in some form in the past (e.g., executing any code from an unknown source either downloaded from a network or supplied on floppy disk). Mobile agents simply offer a greater opportunity for abuse and misuse, broadening the scale of threats significantly.

A number of models exist for describing agent systems (Fuggettaet al, 1998); however, for discussing security issues it is sufficient to use a very simple one, consisting

of only two main components: the agent and the agent platform. Here, an agent is comprised of the code and state of the information needed to carry out some computation. Mobility allows an agent to move, or hop, among agent platforms. The agent platform provides the computational environment in which an agent operates. The platform from which an agent originates is referred to as the home platform, and normally is the most trusted environment for an agent. One or more hosts may comprise an agent platform, and an agent platform may support multiple computational environments, or meeting places, where agents can interact. Since some of these detailed aspects do not affect the discussion of security issues, they are omitted from the agent system model depicted in Figure 1.

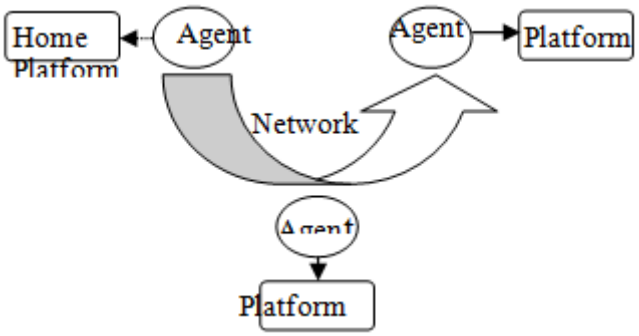

Figure 1: Agent System Model

According to (Alfalayleh and Brankovic, 2003) four threat categories are identified: i). threats stemming from an agent attacking an agent platform, ii). an agent platform attacking an agent, iii). An agent attacking another agent on the agent platform, and iv). Other entities attacking the agent system. The last category covers the cases of an agent attacking an agent on another agent platform, and of an agent platform attacking another platform, since these attacks are primarily focused on the communications capability of the platform to exploit potential vulnerabilities. The last category also includes more conventional attacks against the underlying operating system of the agent platform.

3. The Proposed System

The use of mobile agents in today's network is on the increase and so also the attendant threats to the safety of those agents and messages they carry on their itinerary. There is therefore, a need for a robust scheme that ensures the security of the agents against other malicious agents and against malicious host.

In this work, a security framework is proposed for Mobile Agent Software (MAS) in a distributed computer network. The model uses a combination of two security schemes – the Rivest Shamir Adleman (RSA) algorithm and the Digital Signature Algorithm (DSA). These security schemes offer a better solution in securing Mobile Agent System. The security framework is designed on Java platform as a standard Java desktop (network) that runs on any operating system with the appropriate Java Virtual Machine (JVM) specific to that operating system.

3.1 Background Concepts for the Proposed System

The Communication Scenario: If we assume a network of **n** computers that define the places the agent can visit and if we assume the movement of the agent at a particular point in time on the network is from computer A **(Sender)** to computer B **(Receiver)** as shown in Figure 2:

Figure 2: Communication Scenario

In this two-entity communication scenario, there are so many risks and potential security threats to the Mobile Agent, the message it carries and the host computer. First, the mobile agent code might be intercepted by an enemy and modified along the way and thus the agent code might be unable to initialize itself on the host computer and hence, might not be able to perform assigned task(s). In another way, other malicious agents in-between the communication channel might modify the message(s) carried by the agent. Thus, the agent code itself is not tampered with but the message(s), thus making the agent to perform a different task on the receiving computer. These are usually referred to as **deception** as shown in Figure 2.1(a).More so**,** it is possible for anotheragent (malicious agent) to carry an entirely different code and pretended to be from the original sender and performs some dangerous activities on the host system. This is usually called **masquerading** as shown Figure 2.1(b).

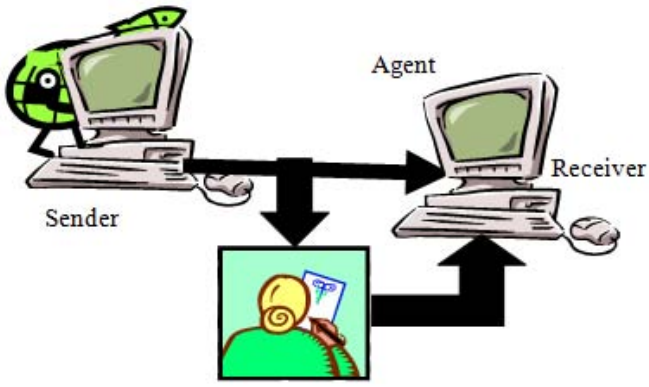

Figure 2.1(a): Deception

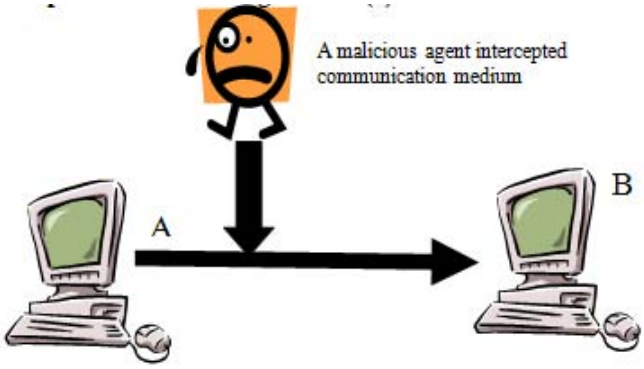

Figure 2.1(b): Masquerading

In the light of the various possible threats to the safety of communication between the sending and receiving computer systems, there is need for a robust security scheme that guarantees solution to these threats and other security issues.

4. Cryptographic Schemes

Various mathematical techniques have been deployed in the implementation of security. In our model, we propose a hybrid of RSA and DSA schemes so as to guarantee the necessary security requirements for a mobile system. Using RSA guarantees confidentiality of data, and data integrity, while Digital signature ensures non-repudiation and data authenticity

4.1 RSA Algorithm

Rivest **S**hamir **A**dleman (Rivest, Shamir and Adleman (1978) algorithm is the most widely used public-key algorithm today. This might be due to its greater level of confidentiality, relatively easy to understand and implement. It supports encryption and decryption. RSA gets its security from factorization problem - difficulty of factoring large numbers (finding number's prime factors) is the basis of security of RSA. The RSA problem assures the security of the RSA encryption and RSA digital signatures.

Operations:

The RSA algorithm involves three steps: key generation, encryption and decryption.

Key generation: RSA involves a public key and a private key. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key. The keys for the RSA algorithm are generated as follows:

1. Select at random, two "large" prime numbers p and q, of approximately equal size such that their product n = pq is of the required bit length, e.g., 1024 bits.
2. Compute modulus n = pq…….. (1)
3. compute phi, Ø = (p-1)(q-1)
4. select an integer e, 1< e < Ø such that gcd (e, Ø) = 1
5. compute private exponent d, 1 < d < Ø, such that ed = 1 (mod) Ø……(2). In other words, $d = e^{-1} \bmod Ø$…… (3).
6. Publish pair P = (e, n) as RSA public key.
7. Do not publish but keep secret pair S = (d, n) as RSA secret key.

The values of p, q, and Ø should also be kept secret.

Where

- n is known as the modulus
- e is known as the public exponent or encryption exponent.
- d is known as the secret exponent or decryption exponent.

4.1.1 Encryption and Decryption

In cryptography, a message (denoted, M) is plaintext (in mobile agent case, this may be the agent code, the message it carries or some other agent information). The technique of disguising a message in such a way as to hide its substance is encryption. An encrypted message is referred to as ciphertext (denoted, C). The process of converting ciphertext back to plaintext is decryption. This is illustrated in Figure 3:

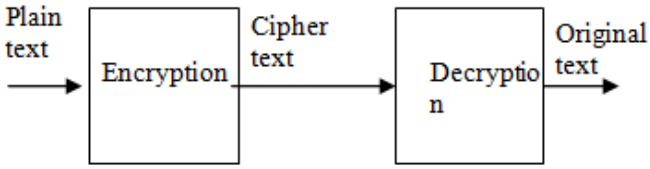

Figure 3: Encryption/Decryption

The encryption function (E) operates on M to produce C. Similarly, the decryption function (D) operates on C to produce M. The encryption and decryption processes are often represented mathematically as:

E (M) = C (encryption)
D(C) = M (decryption)

In most of the cryptographic algorithms, encryption and decryption are carried out using keys. If a different key is used for encryption and decryption, the function could be re-written as:

$E_k(M) = C$ where k is the encryption key
$D_k(C) = M$ where k is the decryption key.

To encrypt a message m, first divideitinto numerical blocks smaller than n. The encrypted message, c, will be made of smaller sized message block, c_i, of about the same length. The encryption formula is simply:

$c_i = m_i^e \bmod n$……(4)

To decrypt a message, take each encrypted block c_i and compute:

$m_i = c^d_i \bmod n$ ……(5)

The message to be encrypted is represented as number m, 0 < m < n -1. If the message is longer it needs to be split into smaller blocks.

4.2 The Digital Signature Algorithm (DSA)

The Digital Signature algorithm (Kitsos et al, 2009) is a technique designed to provide the digital counterpart to a handwritten signature. Digital signatures have many applications in information security, including authentication, data integrity, and non-repudiation. In the mobile agent case in particular, digitally signing the agent code and the messages its carries would guide against malicious agent sending messages and pretending it is from an intended sender. These kinds of messages might be harmful to the host system (receiver) that would run the mobile agent code. A digital signature verification

algorithm is a method for verifying that a digital signature is authentic. A digital signature scheme consists of a signature generation and an associated verification algorithm. A digital signature signing process consists of adigital signature generation algorithm, along with a method for formatting data into messages which can be signed. A digital signature verification process consists of a verification algorithm, along with a method for recovering data from the message.

Suppose X (sender) wishes to send Y (receiver) a digitally signed agent message M.

- X computes its digital signature $s = S_X(M)$ where M is the agent code/message and S_X is the secret key.
- It sends message/signature pair (M, s) to Y.
- Y receives (M, s) and verifies M using $P_X(s)$, where P_X is sender's public key and s is the sender's signature. If the signature is correct, it concludes that M was actually signed by X, otherwise it suspects that M and/or s have been tampered with (forged), or that there were some transmission problems.

Digital signatures provide bothauthentication of signer's identity andauthentication of content of the signed message.

5. Model Architecture

The architecture of the security model for mobile agent is shown in Figure 4. In the architecture, there are four major processes: the encryption process, the signing process, the verification process and the decrypting process. The mobile agent code and the message(s) it carries is first encrypted by the client, transforming the message to an encrypted message; thereafter, the client appends its digital signature to both the encrypted agent and the message so as to authenticate their source at the server's end. The encrypted/authenticated message/code is then sent through the network (usually an untrusted channel) to the receiving end. The server receives the encrypted/authenticated message/code and first, verifies the signature on the agent and the message(s). The server would proceed to allow the agent message to run on its system if and only if the signature is verified to be authentic and if it is from an intended client. If the signature is authentic, the server system then decrypts the agent and its message(s).

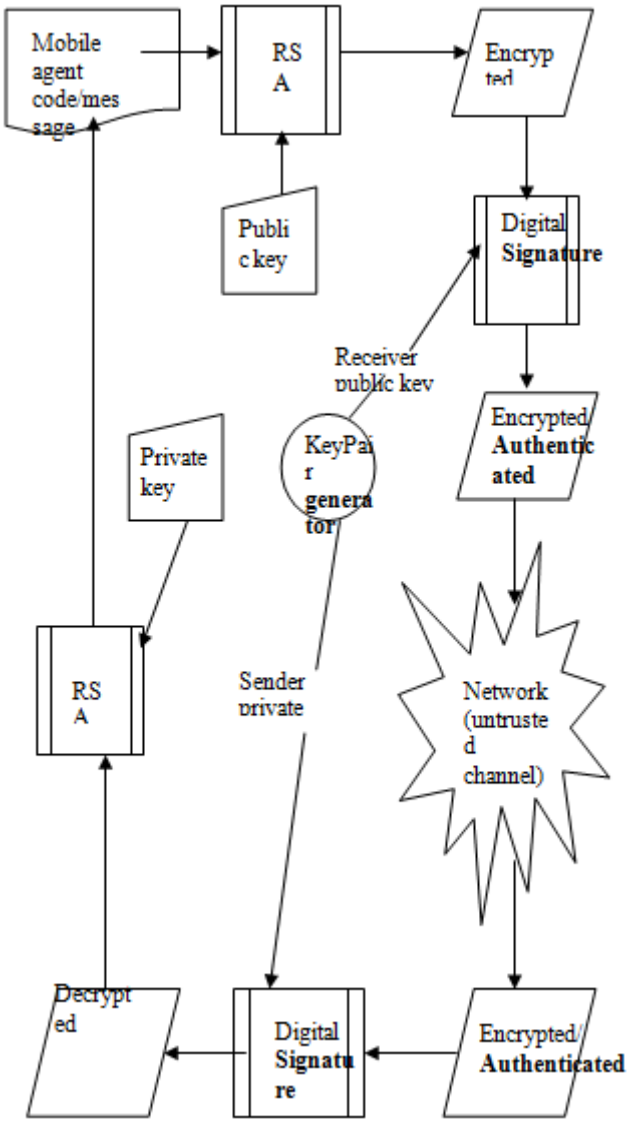

Figure 4: Model Architecture

6. Model Implementation

The proposed model was implemented using NetBeans 6.0 Java programming language. The whole program was implemented in about six modules viz, the login, acknowledgement module, encryption module, signing module, verification module and decryption module. The encryption, signing and acknowledgement processes take place at the client's system while the verification and decryption processes take place at the server's system.The encryption module is implemented based on the RSA encryption scheme as seen in Figure A1. Two keys are generated in the process, which are: the public key (e) that is used for encryption and the private key (d) that is needed for decryption. The private key is needed by receiver to decrypt and so in real life the key is usually assumed to be sent through another means. In this particular case, the decrypting key is stored in a keyfile.txt. To append digital signature to the MA, a KeyPair (public and private keys) is generated as shown in Figure A3. These keys are encoded to guarantee their security. The message is signed once a KeyPair is generated. Actual signing is a four-step process:

1. Acquire a Signature object usingDigitalSignature Algorithm (DSA). The message digest algorithm is SHA-1. For this implementation "SHA/DSA" was used.
2. To sign the message, the Signature is initialized with the Private Key acquired from the generated KeyPair.
3. The Signature is provided with message to sign.
4. Sign the message and get digest. Typically, this would be appended to the original message and sent away to the server.

In this implementation, the signing is done immediately after the MA with its message is ready to migrate to the server's system. The signed file and the key for the signature that would be used for verification are sent to the server. The sample digital certificate is shown in Figure A5.

Verification is only a three-step process:

1. Acquiring a Signature object using DSA.
2. Verifying the data. In so doing the Signature must be initialized with the sender's PublicKey. Normally, this would be acquired from MA generated Certificate.
3. Finally, using the content and the message digest, signature is easily verified.

In the verification module for this implementation, the server supplies the public key filename and the name of the file that was signed as shown in Figure A4. In a case where different key or message was sent by malicious agents on the way, the signature scheme would return an invalid signature upon such message and such a message could be discarded before decrypting as shown in Figure A6.

Upon verification, the message could now be decrypted with the private key generated at the encrypting end. The server receives the message of acknowledgment and decrypts it for verifying the MA (Figure A4). In this implementation, the decrypted message is used to confirm that the MA came from the claimed server. The MA decrypted message is as seen in Figure A7.

7. Summary

In this paper, a model has been developed to tackle some of the security threats that mobile agent systems do face on its itinerary. Particularly, the model has been able to protect mobile agent against malicious agent by implementing a scheme to encrypt the agent code and messages from unintended recipient thereby preventing alteration to the code and the message. The model has been able to safeguard the host machine (receiver) from being attacked by malicious agent ensuring both the agent code and the message(s) are authenticated through a digital signature. The recipient would not be prone to danger if signature is verified authenticated. Information that is not verified successfully may simply be disabled or entirely discarded.

7.1 Conclusion

This work presents a method for securing mobile agents in a distributed system. The basic concepts of agent technology have been introduced. Some important protection mechanisms have been evaluated and discussed. In this model, a hybrid of two security schemes - RSA and DSA have been proposed to guarantee the necessary security requirements for a mobile agent in a distributed system. Using RSA guarantees confidentiality of data and data integrity of the agent; while Digital Signature ensures non-repudiation and data authenticity.

7.2 Future Work

In the future, we intend to look into some other threats to mobile agent's security such as protecting the agent system against malicious host. For example, when the agent system gets to the host machine, the agent might not receive the needed operating environment from the host and thus the agent might fail in its assignment. We would investigate means of preventing the mobile agent system from such an attack.

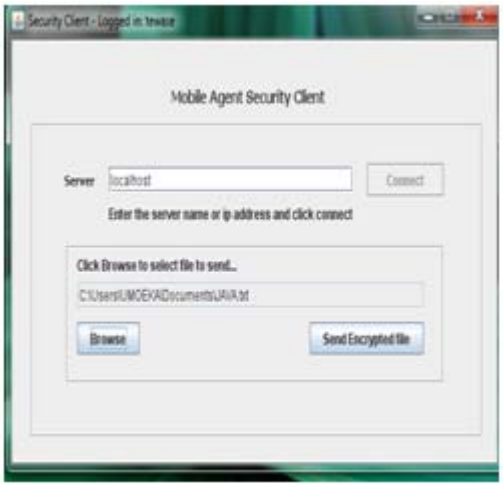

Figure A1: Encryption and Signing Interface

Figure A2: Screen Showing sample of agent encrypted message

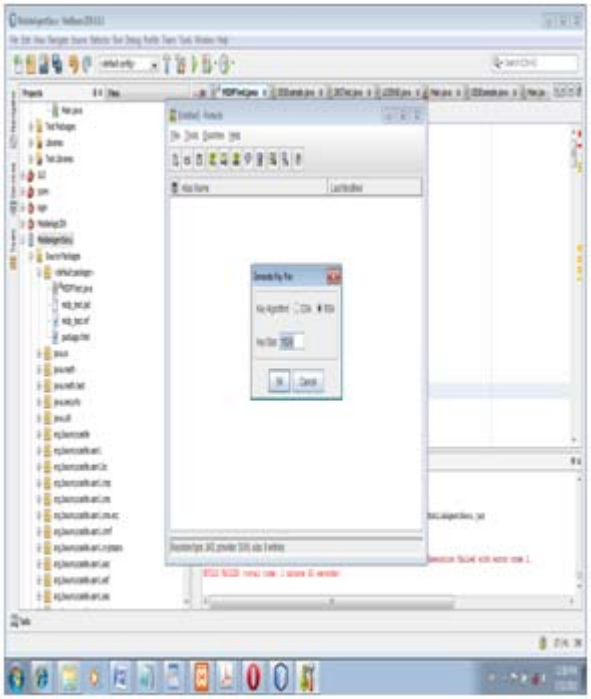

Figure A3: Generation of Key Pair

Figure A4: Verifying and Decrypting interface

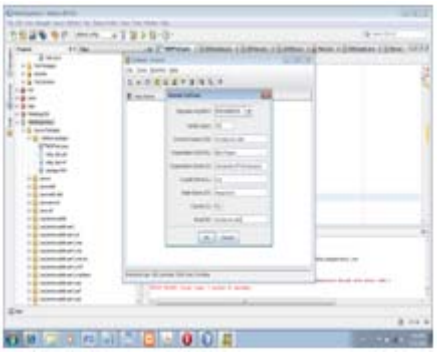

Figure A5: Generation of Certificate

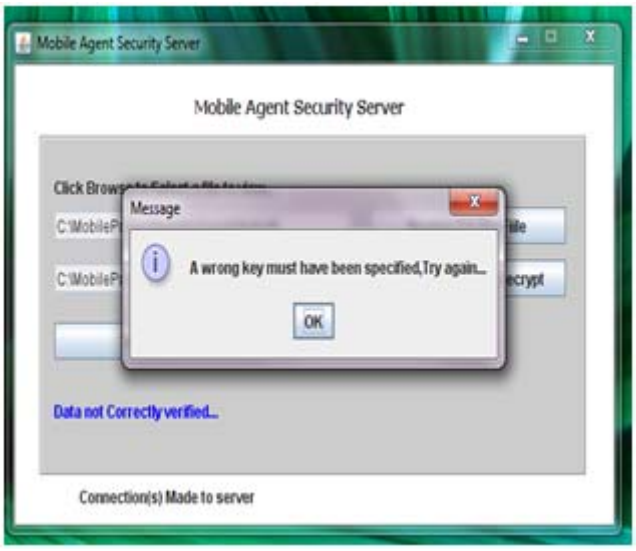

Figure A6: Screen showing interruption by malicious agent

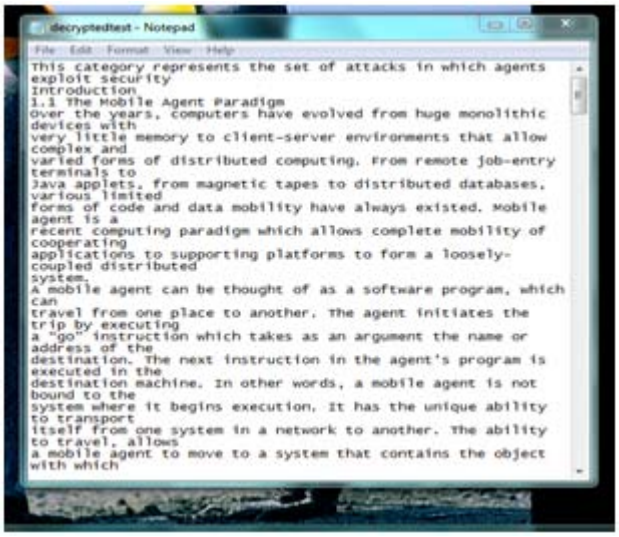

Figure A7: Screen Showing sample of agent decrypted message

References

[1] P. Kitsos, N. Sklavos and O. Koufopavlov (2009): An efficient Implementation of the Digital Signature Algorithm, VLSI Design Laboratory, Electrical and Computer Engineering Department, University of Patras, Patras, Greece, Online Available: http://dsmc.eap.gr/members/pkitsos/papers/Kitsos_c09.pdf

[2] Rivest, R.; Shamir, A.; and Adleman, L. (1978): A method for obtaining digital signatures and public Key Cryptosystems, Communications of the ACM.

[3] Chess, David M. (1998): Security Issues in Mobile Code Systems. Mobile Agents and Security, LNCS Vol. 1419, Springer-Verlagpp1−14

[4] Alfalayleh, Mousa and Brankovic, Lijiljana (2008): An Overview of Security Issues and Techniques in Mobile Agents, The School of Electrical Engineering and Computer Science, The University of Newcastle, Newcastle, Australia, pp. 51-59.

[5] Braun, P.andRossak, W. (2005): Mobile Agents: Concepts, Mobility Models, & the Tracy Toolkit, Elsevier Inc. (USA).

[6] Fuggetta, A., Picco, G.P. and Vigna, G. (1998): "Understanding Code Mobility" IEEE Transactions on Software Engineering, 24(5), http://www.cs.ucsb.edu/~vigna/listpub.html (last visited December, 2012)
[7] Jennings, N. R., and Wooldridge, M. J (1998): editors. Agent Technology, Springer-Verlag
[8] White, J. E. (1995): Mobile Agents. Technical report, General Magic Inc., October 1995