Securing Mobile Agents through the Concept Bioinformatics

Biren Suhag[1]* Dr. Subham Gandhi[2]

[1] Research Scholar, Singhania University, Rajasthan

[2] Research Supervisor, Singhania University, Rajasthan

Abstract – The agent paradigm is currently attracting much research. A mobile agent is a particular type of agent with the ability to migrate from one host to another, where it can resume its execution. In this paper we consider security issues that need to be addressed before multi-agent systems in general, and mobile agents in particular, can be a viable solution for a broad range of commercial applications using the concept of bioinformatics. This is done by considering the implications of the characteristics given to agents and the general properties of open multi-agent systems. The paper then looks in some more detail at security technology and methods applicable to mobile agent systems.

Keyword; Multiagent, Bioinformatics, Security

- X -

1. INTRODUCTION

Agents are independent pieces of software capable of acting autonomously in response to input from their environment. Agents can be of differing abilities, but typically, possess the functionality needed to fulfill their design objectives. To be described as 'intelligent', software agents should have the ability to act autonomously that is without direct human interaction, be flexible, and, in a multi-agent system, be able to communicate with other agents that are to be social. Agents are, to various degrees, aware of their environment, which can also often be affected by the agent's actions.

A mobile agent is a particular class of agent with the ability during execution to migrate from one host to another, where it can resume its execution. It has been suggested that mobile agent technology, amongst other things, can help to reduce network traffic and to overcome network latencies'. An agent's ability to move does however introduce significant security concerns.

The concept of an agent originates from the area of artificial intelligence (At) but has now gained more widespread acceptance in mainstream computer science. The term. 'agent' has become rather fashionable, and a more mature technology than currently available is often implied. This is in particular true for security in multi-agent systems. Oversimplified assumptions and non-applicable references to security solutions are not uncommon in the literature. Naturally, security is not a driving force for research and development of multi-agent systems and therefore has not received much attention from the agent community. Nevertheless, in order for agent technology to gain widespread use and provide viable solutions on a wider scale for commercial applications, security issues need to be properly addressed.

Autonomous agents and multi-agent systems represent a relatively new way of analyzing, designing, and implementing complex software systems. In this paper we are only concerned with the security of the system and its components (leaving design methodologies to others). Several multi-agent systems are available as commercial products and many more have been implemented in various research projects, with varying success. Recent standardization has proven rather successful and are still evolving. Today there is growing interest and research in implementing and rolling out (open) multi-agent systems on a wider scale. The collaborative partnership Mobile VCE is undertaking one such project in which the agent paradigm is being researched in a mobile telecommunications setting.

2. AGENTS AND MULTI-AGENT SYSTEMS

Agents are software entities that have some kind of autonomy and certain 'intelligence'. An agent is often assumed to represent another entity, such as a human or an organization on whose behalf it is acting. No single universal definition of an agent exists, but there are certain widely agreed universal characteristics of agents; these include situations, autonomy, and flexibilities:

- Situatedness means that the agent receives sensory input from its environment and that it can perform actions which change the environment in some way.
- Autonomy means that air agent is able to act without the direct intervention of humans (or other agents), and that it has control over its own actions and internal state.
- Flexibility can be defined using responsive, proactive etc.

Computer hosts, or platforms, provide agents with environments in which they can execute. A platform typically also provides additional services, such as communication facilities, to the agents it is hosting. In order for agents to be able to form a useful open multi-agent system in which they can communicate and co-operate, certain functionality needs to be provided to the agents. This includes functionality to find other agents or to find particular services. This can be implemented as services offered by other agents or services more integrated with the MAS infrastructure itself. Examples of such services include facilitators, matchmakers, mediators, and blackboards.

Open multi-agent systems are usually envisaged as systems, communicating over the Internet, that allow anybody to connect to a platform on which agents are running. This means that the MAS lacks a global system control and that information in general is highly decentralised.

3. SECURITY IMPLICATIONS

This section discusses agent security issues based on the characteristics described in the previous section.

Agent execution

Naturally agents need to execute somewhere. A computer is ultimately responsible for the correct execution and protection of the agent. This leads us to the question of where access control decisions should be performed and enforced. Does the agent contain all the logic and information required to decide whether an incoming request is authentic (originating from its claimant) and, if so, whether it is authorized (whether it has the right to access the requested information or service)? Or can the agent rely on the platform for access control services?

Situatedness

Agents perform their actions while situated in a particular environment. The environment may be a computational one (e.g., a Web site) or a physical one (e.g., a manufacturing pipeline), and an agent can sense and effect some portions it.

Autonomy

Autonomy, when combined with other features given to agents, can introduce serious security concerns. If an agent, for example, is given authority to buy or sell things, it should not be possible for another party to force the agent into committing to something it would not normally commit to. Neither should an agent be able to make commitments it cannot fulfil. Hence, issues related to delegation needs to be considered for agents.

Communication

Of the flexibility properties, social behaviour is certainly interesting from a security point of view. This means that agents can communicate with other agents and humans. Just as an agent's communication with its environment needs to be protected, so does its communication with other agents and humans. The following security properties should be provided such as confidentiality, data integrity, authentication of origin, non-repudiation.

Some implementations of multi-agent systems assume that security is provided transparently by a lower layer. This approach might be sufficient in a closed system where the agents can trust each other and the only concern is external spiteful parties. However, within Mobile VCE we believe that agents in an open system need to be 'security aware', i.e. they need to be able to make decisions based on where information is originating from and how well protected it is. As suggested elsewhere public key cryptography using the concept of bioinformatics and a supporting public key infrastructure can be used as important parts of inter-agent communication.

Mobility

Agents need protection from other agents and from the hosts on which they execute. Similarly, hosts need to be protected from agents and from other parties that can communicate with the platform. The problems associated with the protection of hosts from spiteful code are quite well understood.

The problem posed by spiteful hosts to agents seems more complex to solve. Since an agent is under the control of the executing host, the host can in principle do anything to the agent and its code. The particular attacks that a spiteful host can make can be observation, data and flow control, manipulation, manipulating the route of an agent, incorrect execution of codes, including re-execution, denial of execution, either in part or whole, masquerading as a different host, eavesdropping on agent communications

Rationality, authenticity, and goodwill

'Agents are well behaved and will never act in a spiteful manner.' If we make this a true requirement, then the required redundancy for such a system is likely to make the system useless. It would, of course, be valuable to have a system in which agents can be assumed to behave truthfully in every situation. This does not seem a likely scenario for a multi-agent system that is not under very strict control and under a single authority, and would not correspond to the assumed open system scenario.

Identification and authentication

An agent could simply be identified by something like a serial number, or its identity could be associated with its origin, owner, capabilities, or privileges. If identities are not permanent, security-related decisions cannot be made on the basis of an agent's identity.

Authorisation and delegation

Authorisation and delegation are important issues in multi-agent systems. Agents need to be granted rights to access information and other resources in order to carry out their tasks; they also act on behalf of a person, organisation, or other agents, requiring transfer of access rights between different entities.

4. SECURITY MEASURES FOR MOBILE AGENTS

It is common for multi-agent system implementations to assume a virtual private network underlying network for providing security services. This approach usually does not provide much flexibility, since secure communication between parties without pre-established relationships becomes unwieldy. Nevertheless, this solution can use well established security protocols and be adequate for applications in which a communication is protected to the same degree. Such an approach usually leaves the agents completely unaware of security services, as this is handled between agent platforms (or perhaps even at the link level). The agents themselves are also unprotected from spiteful hosts if no other security measures are applied. Few available technologies, Mechanisms addressing various security aspects of the mobile agent are considered first, and then technologies protecting the executing hosts from agents are examined like

Protecting agents

The security issues that arise with mobile agents are rather well understood by the security community and hence much research is being devoted to resolving them. Most of the many attempts to counter the threats posed to mobile agents have addressed a particular part of the problem.

As stated before, once an agent has arrived at a host, little can be done to stop the host from treating the agent as it likes. The problem is usually referred to as the spiteful host problem. A simple example often used to illustrate how a spiteful host can benefit from attacking a mobile agent is the shopping agent. An agent is sent out to find the best airfare for a flight with a particular route. The agent is given various requirements, such as departure point, destination and time restrictions, and sent out to find the cheapest ticket before committing to a particular purchase. The agent will visit every airline and query its databases before committing to a purchase and reporting back to the agent owner. A spiteful host can interfere with the agent's execution in several ways in order to make its offer appear the most attractive. For example, a spiteful host could try: (a) to erase all the information previously collected by the agent, so that the host is guaranteed at least to be making the best current offer; (b) to change the agent's route so that airlines with more favourable offers are not visited; (c) simply to terminate the agent to ensure that no competitor gets the business either; (d) to make the agent execute its commitment function, ensuring that the agent is committing to the offer given by the spiteful host (if the agent is carrying electronic money the spiteful host could instead take it from the agent directly). In addition to this, the agent might be carrying information that needs to be kept secret from the airlines (e.g. maximum price).

There is no universal solution to the spiteful host problem, but some partial solutions have been proposed. Many of the security mechanisms are aimed at detecting, rather then preventing, misbehaving hosts. The following subsections describe some of the mechanisms that have been proposed to address die spiteful host problem.

Contractual agreements

The simplest solution (at least from a technical perspective) is to use contractual means to tackle the spiteful host problem. Operators of agent platforms guarantee, via contractual agreements, to operate their environments securely and not to violate the privacy or the integrity of the agent, its data, and its computation. However, to prove that such an agreement has been broken might be a non-trivial task.

Trusted hardware

If the operators of the available execution environments cannot be trusted, one obvious solution is to let a trusted third party supply trusted hardware, in the form of tamper-resistant devices, that are placed at the site of the host and interact with the agent platform'. A tamper-resistant device can, for example, come in the form of a smart card. Such trusted hardware can then either protect the complete execution environment of the agent or perform certain security-sensitive tasks. However, such trusted hardware must be used carefully and might appear to offer more security than it really does. The agent must still be able to communicate with resources at the local platform (the part under the control of an untrusted party), for example to interact with a local database. All such interactions can still be intercepted by the untrusted party.

If the trusted hardware is only used to protect security-sensitive actions this might be even more vulnerable. It might, for example, be tempting to let the agent's private signature key be protected such that it will only be available when decrypted inside the trusted device. A signature algorithm can then be executed within the device using the agent's private key. In this way, the private signature key is never exposed to the host. However, the host might be able to interfere with the communication taking place between the agent residing on the host and the trusted device in such a way that a correct signature is produced on information falsely manufactured by the host.

Above all else, the major drawback of trusted hardware is the cost of such a solution.

Trusted nodes

By introducing into the infrastructure trusted nodes to which mobile agents can migrate when required, sensitive information can be prevented from being sent to untrusted hosts, and certain misbehaviours of spiteful hosts can be traced. The owner's host, i.e. the platform from which the mobile agent was first launched, is usually assumed to be a trusted node. In addition to this, service providers can operate trusted nodes in the infrastructure. In our example with the shopping agent, the mobileagent can be constructed so that the commitment function (e.g. the agent's signature key) is encrypted such that it can only be decrypted at a trusted host. Once the agent arrives at the trusted host it can compare the collected offers and commit to the best offer. Alternatively9, one agent containing the ability to commit to a purchase can be sent to a trusted node. rrom this node one or several sub-agents are sent to the airline hosts to collect offers. Depending on the threat scenario, single-hop agents can be used—that is agents visiting only one host before returning—or one or several multi-hop agents can be used. Once the sub-agent or agents have returned to the trusted node, the best offer is selected and the agent commits to a purchase. This last alternative does limit the agent's mobility, but may be beneficial in certain scenarios.

Co-operating agents

By using co-operating agents, a similar result to that of trusted nodes can be achieved'('. Information and functionality can be split between two or more agents in such a way that it is not enough to compromise just one (or even several) agent in order to compromise the task. An identical scenario to that described using trusted nodes can, for example, be achieved by letting the agent residing on the trusted host be executed on any host that is assumed not to be conspiring with any of the airlines.

By applying fault-tolerant techniques the spiteful behaviour of a few hosts can be countered. One such scheme for ensuring that a mobile agent arrives safely at its destination has been proposed in Reference 11. Although a spiteful platform may cause an agent to operate incorrectly, the existence of enough replicates ensures the correct end result.

Again, referring to the shopping agent, several mobile agents can be used, taking different routes, and before deciding on the best offer the agents communicate their votes amongst each other.

Execution tracing

Execution traCing12 has been proposed f-)r detecting unauthorised modifications to an agent through the faithful recording of the agent's execution on each agent platform. Each platform is required to create and retain a non-repudiable log of the operations performed by the agent while executing on the platform. The major drawbacks of this approach are not only the size of the logs created, but also the necessary management of the logs created.

Partial result authentication codes (PRACs) were introduced byYee13. The idea is to protect the authenticity of an intermediate agent state or partial result that results from running on a server. PRACs can be generated using symmetric cryptographic algorithms. The agent is then equipped with a number of encryption keys. Every time the agent migrates from a host, the agent's state or some other result is processed using one of the key-, to produce a message authentication code (MAC) on the message. The key that has been used is then disposed of before the agent migrates. The PRAC can be verified at a later point to identify certain types of tampering. A similar functionality can be achieved using asymmetric cryptography by letting the host produce a signature on the information instead.

Encrypted payload

Asymmetric cryptography (also known as public key cryptography) is well suited fora mobile agent that needs to send results back to its owner or that collects information along its route before returning to its owner with its encrypted payload. This is due to the fact that the encryption key does not need to be kept secret. However, to encrypt very small messages is either very insecure or results in a large overhead compared with the original message. A solution called sliding encryption" has been proposed that allows small amounts of data to be encrypted and consequently added to the cryptogram, but still yields efficient results in terms of size. Due to the nature of asymmetric cryptography the agent is not able to access its own encrypted payload until arriving at a trusted host where the corresponding decryption key is available.

Environmental key generation

Environmental key generation's allows an agent to carry encrypted code or information. The encrypted data can be decrypted when some predefined environmental condition is true. sing this method an agent's private information can be encrypted and only revealed to the environment once the predefined condition has been met. This requires that the agent has access to some predictable information source. Once the private information has been revealed, it would, of course, be available also to the executing host. However, if the condition is not met on a particular host, the private information is not revealed to the platform.

Computing with encrypted functions

Sander and Tscbudin16 have proposed a scheme whereby an agent platform can execute a program embodying an enciphered function without being able to discern the original function. For example, instead of equipping an agent with function f, the agent's owner can give the agent a program P(E(f)) that implements E(f), an encrypted version of f. The agent can then execute P(E(f)) on x, yielding an encrypted version of f (x).

With this approach an agent's execution would be kept secret from the executing host, as would any information carried by the agent. For example, the means to produce a digital signature could thereby be given to an agent without revealing the private key. However, a spiteful platform could still use the agent to produce a signature on arbitrary data. Sander and Tschudin therefore suggest combining the method with undetachable signatures (see below).

Although the idea is straightforward, the trick is to find appropriate encryption schemes that can transform functions as intended; this remains a research topic. Recently Barak et al.11 have shown that it is unlikely that this approach will succeed.

Obfuscated code

Hohll8 proposes what he refers to as Blackbox security to scramble an agent's code in such a way that no one is able to gain a complete understanding of it-, function. However, no general algorithm or approach exists for providing Blackbox security. A time-limited variant of Blackbox protection is proposed as a reasonable alternative. 'this could be applicable where an agent Deeds to be protected for a short period only. One serious drawback of this scheme is the difficulty of quantifying the protection time provided by '.he obfuscation algorithm.

Undetectable signatures

By binding usage restrictions to a signature key given to the agent, we can potentially limit the damage that a spiteful host can do. Sander and TSCHUDIN (3 proposed one such scheme, which they refer to as undetachable signatures. Their original scheme has since been improved'. The idea is to encode constraints into the signature key. If the constraints are not met a valid signature is not produced, preventing arbitrary messages from being signed.

An alternative to undetachable signatures is to use digital certificates to regulate the validity of digital signatures. Digital certificates are used to let a verifier check the validity of a digital signature. Certificates usually include a validity period during which valid signatures can be produced. By extending the constraints included in the certificate to context-related values such as executing host, maximum value of a purchase, and so on, certificates can be used to restrict further the usage of signature keys and thereby decrease the risks involved regarding improper use of the signature key. One advantage of this scheme over undetachable signatures is that it relies on already well-established cryptographic techniques.

Protecting the agent platform

More mature technology is available to address the problem of protecting the agent platform from spiteful agents. Techniques similar to those used to address security issues associated with downloading software from the Internet can be applied to the mobile agent scenario.

Sandboxing and safe code interpretation

Sandboxing isolates applications (or in our case agents) into distinct domains enforced by software. This technique allows untrusted programs to be executed within their virtual address space, thereby preventing them from interfering with other applications. Access to system resources can also be controlled through a unique identifier associated with each domain.

Agent-, are usually developed using an interpreted script or programming language. The main motivation for this is to support agent platforms on heterogeneous computer systems. The idea behind safe code interpretation is that commands that are considered insecure can be either made safe or denied to the agent. Java is probably the most widely used interpretative language used today. Java also utilises sandboxing and signed code (described below); this makes it well suited for the development of mobile agents.

Proof-carrying code

Proof-carrying code 'O requires the author of an agent to formally prove that the agent conforms to a certain security policy. The execution platform can then check the agent and the proof before executing the agent, which can then be run without any further restrictions. The major drawback of this approach is the difficulty in generating such formal proofs in an automated and efficient way.

Signed code

By digitally signing an agent its authenticity, origin and integrity can be ensured. Typically the code signer is either the creator of the agent, the agent owner (on whose behalf the agent is acting), or some party that has reviewed the agent. The security policy at the platform, perhaps in conjunction with attribute certificates supplied with the signed code, would then decide if a particular signature means that the code should be executed.

Path histories

The idea behind path histories is to let a host know where a mobile agent has been executed previously. If the agent has been running on a host that is not trusted, the newly visited host can decide not to let the agent execute or restrict the execution privileges. Path histories require each host to add a signed entry to the path, indicating its identity and the identity of the next platform to be visited, and to supply the complete path history to the next host.

State appraisal

State appraisal" attempts to ensure that an agent's state has not been tampered with and that the agent will not carry out any illegal actions through a state appraisal function which becomes part of the agent code. The agent author produces the appraisal function which is signed, by the author, together with the rest of the agent. An agent platform uses the function to verify that an incoming agent is in a correct state and to determine what privileges an agent can be granted during execulion. The theory, which still is to be proven in practice. requires that the legal states can be captured and described in an efficient and secure way.

5. CONCLUSIONS

The security issues for non-mobile agents can, at least in theory, to a great extent be tackled through existing security technology and protocols. Issues related to trust and delegation in a large-scale multi-agent system are non-trivial to solve. Although a public-key infrastructure is likely to be an important part of the solution, agents need to be able to reason and make decisions based on various security parameters. Execution of agents (mobile as well as non-mobile) on untrusted platforms is another factor introducing non-trivial security concerns, in particular related to correct agent execution and confidentiality of agent data. The required security level and security measures must, as always, depend on the application. Current standardization efforts and large-scale agent projects are likely to facilitate greater use of agent technology in the future. One such research effort is ongoing within the Mobile VCE Core 11 research programme, which has recently proposed a security architecture for agent-based mobile middleware.

There seems to be no single solution to the security problems introduced by mobile agents unless trusted hardware is introduced, which is likely to prove too expensive for most applications. The way forward appears to lie in a range of mechanisms aimed at solving particular (smaller) problems. This could, for example, include mechanisms that depend on agents executing on several hosts rather than on only one host, mechanisms and protocols binding agent actions to hosts, generation of various types of audit information that can be used in case of disputes, and so on. Solutions to certain problems do exist, but for mobile agents to be more widely adopted this is an area that requires further research.

REFERENCES

Barak, B., Goldreicii, O., Impagliallo, R., Rudich, S., Sahai, A., Vadhan, S., and Yang, K. (2001). 'On the (im) possibility of obfuscating programs', in Kilian, J. (Ed.): 'Proceedings of the 21st Annual International Cryptology Conference', Vol. 2139 in LNCS, (Springer-Verlag, Berlin, 2001), pp. 1-18

Borselius, N., Mitchell, C. J., and Wilson, A. T. (2001). 'On, mobile agent based transactions in moderately hostile environments', in DE Decker, B., Piessens, E, Smits_ J., and VAN Herreweghen, E. (Eds.) (2001). Advances in network and distributed systems security'. Proc. IFIP TC11 WGLIA First Annual Working Conf. on Network Security, KU Leuven, Belgium, November 2001, (Kluwer Academic Publishers, Boston, 2001), pp. 173-186

Burg, B. (2000). Towards the deployment of an open agent world'. In Journ6es Francophones d'hitelligence Artificielle Distribu6c et de Systemes Multi-Agents (JFLADSMA2000), 2nd-4th October 2000 (Editions Hermes Science, Paris, 2000)

Genfsereth, M., and Fikes, R. (1992). 'Knowledge interchange format, version 3.0 reference manual'. Technical Report Logic-92-1, Computer Science Department, Stanford University, USA, 1992

Harrison, C. G., Chess, D. M., and Kershenbaum. A. (1995). 'Mobile agents: are they a good idea?' IBM Research Division technical report, 1995.

Hohl, E. (1998). Time limited blackbox security: protecting mobile agents from spiteful hosts', in VIGNA, G. (Ed.): 'Mobile agents and security', Vol. 1419 in LNCS, (Springer-Verlag, Berlin, 1998), pp. 92-113.

Jennings, N. R., and Wooldridge, M. (1995). 'Intelligent agents: theory and practice', Knowl. Eng. Rev., 1995, 10, (2), pp. 115—152

Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). 'A roadmap of agent research and development', Auton. Agents Multi-Agent Syst., 1, (1), pp. 275-306

KO.17 Anikolaou, P, Burme. FER, M., and Chrissikopoulos, V.: 'Secure transactions with mobile agents in hostile environments', in Dawson, E., Clark, A., and Boyd, C. (Eds.): 'Information security and privacy: Proceedings of the 5th Australian Conference-ACISP 2000', Vol. 1841 in LNCS, (Springer-Verlag, Berlin, 2000), pp. 289-297

Luck, M., and D'inverno, M. (2001). 'A conceptual framework for agent definition and development', Comput. I (UK), 44, (1), pp. 1-20

Mahajan R., Gupta T., Mahajan S, Bawa N., Retina as Authentication Tool for Covert Channel Problem, World Academy of Science, Engineering and Technology 56, pp. 153-158

Necula, G. C., and Lee, P. (1998). 'Safe, entrusted agents using proof-carrying code', in VIGNA, G. (Ed.):'Mobile agents and security', Vol. 1419 in LNCS, (Springer-Verlag, Berlin, 1998), pp. 61-91.

QI He, Sycara, K. P., and Finin, T. W. (1998). 'Personal security agent: KQMI, based PKI', in SYCARA, K. R, and Wooldridge, M. (Eds.): 'Proceedings of the 2nd International Conference on Autonomous Agents' (ACM Press, New York, USA, 1998), pp. 377-384

Riordan, J., and Schneier, B. (1998). 'Environmental key generation towards clueless agents, in VIGNA, G. (Ed.): 'Mobile agents and security', Vol. 1419 in LNCS, (Springer - Verlag, Berlin, 1998), pp. 15-24

Roth, V. (1998). 'Secure recording of itineraries through cooperating agents'. Proc. ECOOP Workshop on Distributed Object Security. and 4th Workshop on Mobile Object Systems: Secure Internet Mobile Computations, (INRIA, France, 1998), pp. 147-154.

Sander, T, and Tschudin, C. (1998). Towards mobile cryptography'. Proc. IEEE Symp. on Security and Privacy, Oakland, CA, USA, May 1998, pp. 215-224

Schneider, E B.: Towards fault-tolerant and secure agentry', in Mavronicolas, M., and TSIGAS, P. (Eds.): 'Eleventh International Workshop on Distributed Algorithms', Vol. 1320 in LNCS, (Springer-Verlag, Berlin, September 1997), pp.1-14

VIGNA, G.: 'Protecting mobile agents through tracing'. Proc. 3rd ECOOP Workshop on Operating System Support for Mobile Object Systems, Finland, June 1997, pp. 137-153

Wilhelm, U. G., Staamann, S., and Bu17yan, L. (1999). 'Introducing trusted third parties to the mobile agent paradigm', in Vitek, J., and Jensen, C. (Eds.): 'Secure Internet programming'; Vol. 1603 in LNCS, (Springer-

Verlag, New York, NY, USA, 1999), pp. 471-491.

Workshop on Fast Software Encryption, FSE'97, 20th-22nd January 1997, Haifa, Israel, Vol. 1267 in LNCS, (Springer Verlag; Berlin, 1997)

YEE, B. (1999). 'A sanctuary for mobile agents', in Vitek, J., and Jensen, C. (Eds.): 'Secure Internet programming', Vol. 1603 in LNCS, (Springer-Verlag, New York, NY, USA, 1999), pp. 261-274

Young, A., and Yung, M.: 'Sliding encryption: a cryptographic tool for mobile agents'. Proc. 4th Int.

Corresponding Author

Biren Suhag*

Research Scholar, Singhania University, Rajasthan