Efficient and Secure Web Services by using Multi Agents

Abolfazl Esfandi
Department of Computer Engineering
Islamic Azad University
Boroujerd Branch, Iran

ABSTRACT
Mobile agents are an excellent technology for implementing Web services. Within a set of federated Web services, mobile agents can reduce bandwidth requirements and mitigate the effects of high-latency network connections. This paper presents a model for implementing Web services with mobile agents where agents are free to move between cooperating Web servers to implement the service functionality.

Also for increasing security of web services, we illustrate a novel distributed protocol for multi agent environments. In this approach, the encrypted private key and the message are broken into different parts carrying by different agents, which make it difficult for malicious entities to mine the private key for message encryption, while the private key for the encrypted key is allocated on the predetermined destination nodes. On the other hand, all of the previously proposed encryption algorithms can be applied in the proposed approach that deteriorates the key discovery process. To improve the overall security, the paper makes use of Advanced Encryption Standard (AES) as the encryption base for message encryption.

Our mobile agent Web services present typical WSDL interfaces, so mobile agent functionality can be consumed from legacy clients, and federated services can be gradually migrated to a mobile agent implementation.

General Terms
Multi agent, Cryptographic Protocol, Mobile Agents, Security, and Web services et. al.

Keywords
Multi agent, Cryptographic Protocol, Mobile Agents, Security, Web services, implementation web services by mobile agent.

1. INTRODUCTION
A mobile agent is a discrete bundle of program code and data that can move between hosts. A mobile agent is somewhat like an object, in that has identity and state. A mobile agent is also like a process, in that it contains threads of execution and encapsulates data within a definite. Because mobile agents have threads and program code, they tend to be autonomous entities, unlike relatively passive objects that have methods invoked on them.

A mobile agent can move to a host and then interact with it locally. Interacting locally:

- Reduces bandwidth requirements because an agent can filter data at the source that would otherwise have to be marshaled over the network.
- Moderates the effects of high latency, since there are only agent send- and return-trips.
- Is robust in intermittently connected networks.

These good features are in contrast to those of remote procedure calls (RPC) and Java RMI [5]. Mobile agents also imply support for mobile code and concurrency, which are powerful capabilities.

Because of their good features, we would like to leverage the advantages of mobile agents in implementing Web services. The contribution of this paper is to propose a model of implementing Web services using mobile agents, where:

- Mobile agents are annotated with metadata to describe services.
- Agent life-time is bound to a service request.
- Communication is restricted to unify inter-agent communication and Web service invocations.

Beside, Security is one of the most challenging issues particularly in networked environments because of dispersed number of system users. In traditional security methods, discovering the determined private key is enough for message decryption that can be done through malicious attacks to the network nodes or listening to communication links [15]. The proposed approach, using the multi agent for implementation web service and for improves private key security using two strategies:

- Encrypting the private key using an encryption algorithm (AES algorithm is used in our protocol [2]).
- Breaking the encrypted private key into different units.

Figure 1 generally illustrates the structure of a source node in a networked environment using the proposed strategy. The process loop presented in the figure, starts with the original message and describes all the intermediate processes, which ends toward outgoing links [1].

The paper is organized as follows:

The next section we will talk about basic concepts of distributed protocol for multi agent cryptography. Then explain the implementation and integrating of web service by using multi agent and how mapping web service requests to mobile agents. In section 4 we present my proposed approach and resulted simulation and Finally, evaluation and resulted advantages of our approach are presented.

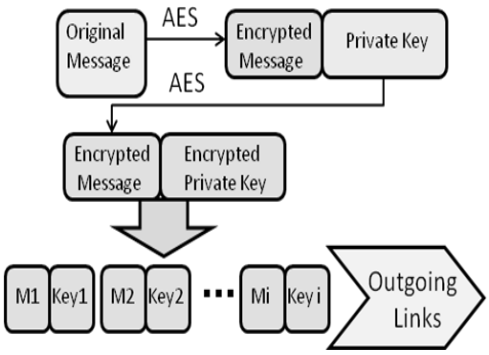

Fig 1: Block diagram of a source node structure which uses the proposed protocol

2. MULTI AGENT ENCRYPTION PROTOCOL

In multi agent systems malicious behaviors should be well studied for better security provisions. A malicious attack against a multi agent system can be considered as one of the following points [4]:

1. Mobile agent security against other malicious mobile agents.
2. Mobile agent security against malicious hosts.
3. Host security against malicious agents.

The proposed approach focuses on the second and the third concept while the first one is not considered in this paper.

Our algorithm improves the overall security by splitting the private key into several parts at the original node and reassembling them at the destination node. The n segregated parts are transferred towards the desired destinations using n mobile agents. According to figure 1 the original message P, is encrypted using a predetermined encryption algorithm $C_{x,y}$ as:

$$\psi = C_{x,y}(P) \quad (1)$$

Where x,y correspond to the public and private keys of the encryption algorithm respectively and ψ corresponds to the encrypted message at the source node. To improve the security the private key y is encrypted again.

The next step is splitting the message and the key into n parts. Splitting of the message and key is accomplished through a Split function, which can be explained as:

$$\psi_i = \text{Split}_i(\psi) \quad (2)$$

The above function is run twice, once for splitting the original encrypted message and once for splitting the private key (ψ_i corresponds to the i^{th} part of the encrypted message).

The source node generates the required number of mobile agents and equips each agent with the predetermined part of the message and the key. It is supposed that the i^{th} agent carries the i^{th} part of the message and the ith part of y'_i (Si corresponds to set of ψ_i and y'_i). This process is illustrated in Figure 2.

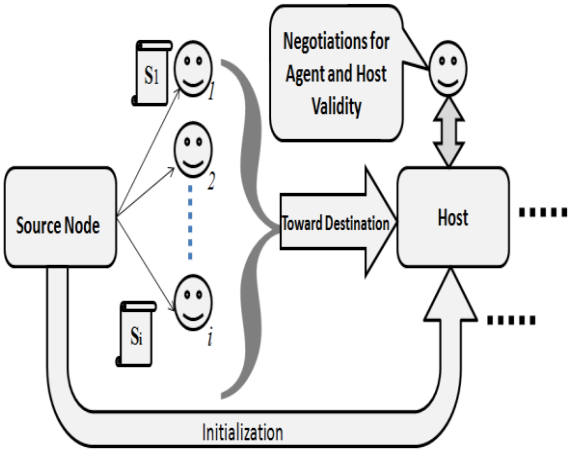

Fig 2: mobile agent's transmissions from source to destination

The destination hosts have some initialization data for this protocol. This initialization data contains λ (private key for key decryption) and the initial state X_0. Origin (source node) signs each mobile agent to avoid malicious behaviors on the network and sends it through the network towards the destination host.

At each host, each agent checks the host's trustworthy using its internal function. On the other hand, each host checks the agent according to digital signatures.

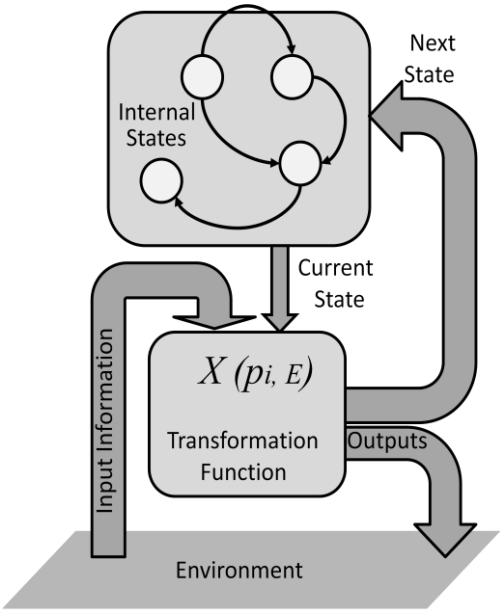

Fig 3: A mobile agent structure and its internal states

As indicated in figure 3 the internal function can be considered as:

$$X(Pi, E) = Pi+1 \quad (3)$$

Where P_i is the current state of the mobile agent, E is used for the environmental parameters and $_{Pi+1}$ indicates the next state of the agent according to the internal manipulations.

We eliminated the role of E in our system and focused on the agents' states. Each mobile agent gets P_{i-1} from each host, calculates the next state P_i and compares the resulted P_i with its internal current state (P_i). The equality indicates that the host is reliable for the other manipulations.

It should be noted that at each host four different states may be determined for the agent. Figure 4 describes these four states together with the related relations.

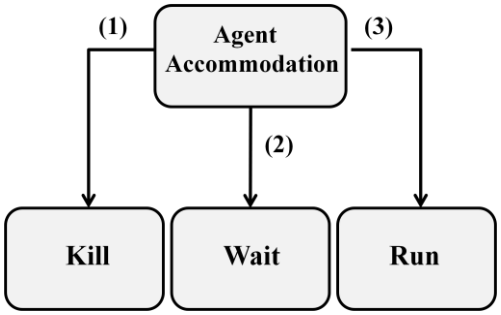

Fig 4: Four states of each agent on a network host

The following text, expresses completely the relations mentioned in figure 4:

1. If the previous agent has not accommodated on the host yet the agent will kill itself. Such a host can be considered as an unreliable host.
2. The previous agent is still working on the host. The arriving agent should wait until complete action of the previous agent.
3. The previous agent has left the host; the arriving agent can run the internal function.

Finally at the destination node a Join function acts conversely to reassemble the encrypted message:

$$\psi = \text{Join}(\psi_i) \quad (4)$$

It should be noted that the integration of the ψ_i is according to their initial segregation orders. The key is also reassembles similar to the message according to (4).

The result of this process is the encrypted key that should be decrypted using the initialized private key and obtained y. finally ψ decrypted by private key y and destination host will obtained original message.

Table 1. Characteristic of the three releases of AES algorithm

	Key Length (Nk words)	Block Size (Nb words)	Number of Round (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

In this protocol, we use AES encryption algorithm for cryptography, AES encryption algorithm is a powerful algorithm introduced by Daemen and Rijmen [3].

This algorithm has three releases, which are illustrated in table 1. The AES algorithm runs four functions namely: SubBytes, ShiftRows, MixColumns and AddRoundKey to complete the encryption process. All of the different classes of AES use these four functions.

AES-128 runs this sequence for 10 times. AES-192 runs for 12 and AES-256 runs the functions for 14 times. As table 1 shows these three releases generate different key sizes. Details about AES encryption algorithm can be found in [2].

3. INTEGRATING WEB SERVICES AND MOBILE AGENTS

In implementing Web services with mobile agents, some issues are quintessential to one discipline or the other. For example, the granularity of the unit of mobility is exclusively a mobile agent-related issue, while marshalling data to and from SOAP messages is exclusively a Web service-related issue.

Other issues, such as security or transactions, are common to Web services and mobile agents. Addressing these issues will involve integrating the work in each area into a consistent whole. This section describes three design issues that exist at the boundary of mobile agents and Web services: service provision, request-agent mapping, and agent communication.

3.1 Service Provision

Traditionally Web services are static and relatively long-lived at a network endpoint. In contrast, a mobile agent may exist only fleetingly on a particular host. The first question in providing Web services with mobile agents is how a service is described. Web Services Description Language (WSDL) [6] describes services as operations on messages at a particular network end-point, with bindings to concrete protocols and message formats.

It would be possible to simply require the programmer to provide a WSDL description of any mobile agents. How requests to the service described in the WSDL are mapped to a mobile agent [10].

In our proposal implementation, the methods of a class that the programmer annotates with the Web Method metadata attribute form the description of a Web service. This is the same way a programmer nominates a method to be part of an ordinary. No other special metadata is required, nor does the programmer have to extend a particular type.

Services are deployed by means of another, ordinary Web service called Agent Service (see Figure 5). Agent Service accepts and loads an assembly and Web service stubs are created for any types that contain service methods. In a single assembly there may be several types containing service methods. Although the types may be related, our implementation publishes them as separate, independent services. Clients then reference these services directly[11].

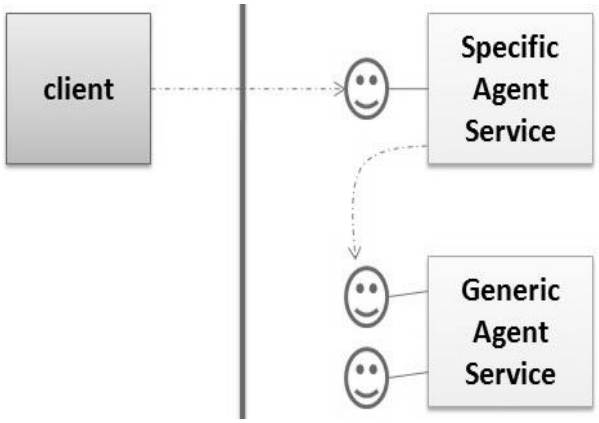

Fig 5: Agent Service interfaces

An alternative method of service provision would be to allow agents to provide and revoke 'ad-hoc' services by identifying a network endpoint and a bundle of methods that requests should be dispatched to. This meshes well with the dynamic flavor of mobile agent-based computing, however it is at odds with web services that provide stable interfaces at a given endpoint [13].

3.2 Mapping web service requests to mobile agents

Like an object in object-oriented programming, a mobile agent is an identifiable, discrete unit. The problem this introduces is one of mapping a Web service request to a particular mobile agent that fulfils that request. Because agents and objects both have a concept of identity, this problem is analogous to that of delivering service requests to object instances in services with object-oriented implementations.

For example, Apache SOAP deployment descriptors [7] can map Web service requests to Java objects that have a per-request, per-HTTP session (if the underlying SOAP transport is HTTP), or per-application lifetime. Furthermore, individual EJB stateful session beans can be addressed via unique URNs. Clients discover these URNs when they call the bean's create method [9].

We can use a one-to-one mapping between services and mobile agents. In this scheme, all requests to a service are delivered to one 'singleton' agent. This single request, single agent model is complex because of the plethora of design choices for when the mobile agent moves away from the service endpoint. Requests could be queued or forwarded to the mobile agent's new location, the service could be torn down, or a new mobile agent could be instantiated to replace the old.

An alternative is to create a new mobile agent for each incoming request. This will limit the lifetime of a particular mobile agent to the length that a request takes to fulfill. However it is a simple model because multiple, concurrent requests can easily be handled by creating more mobile agents. In this scheme, the viability of the Web service is not threatened if a particular mobile agent moves away[8].

3.3 Inter-agent communication

Although inter-agent communication is ostensibly a mobile agent issue, the introduction of another communication mechanism (that is, the Web service method call) may make it desirable to revisit the inter-agent communication with the goal of unifying the mechanisms[14].

However, there is a firm requirement that a Web service implemented with mobile agents should potentially look like any other Web service, and this is at odds with the inter-agent communication mechanisms of some mobile agent systems, such as communication via a shared tuple space [12,16].

It is also reasonable to expect that a mobile agent may want to consume a Web service. It would be useful if legacy Web services could be surfaced to appear to the mobile agent developer merely as another mobile agent, albeit a long-lived, non-moving one.

4. OUR PROPOSED APPROACH AND SIMULATION

In section 2, we present new approach to encrypting message by multi agents, then in section three descript how can mapped web services on mobile agents. Therefore, we use these topics to implement Efficient and secure Web Services by using multi agents.

In the prototype implementation, it is the stub Web service generated by the Agent Service that accepts incoming SOAP messages. The stub delegates to an agent host, which handles the task of directing the request to a particular mobile agent. The agent host does this by first creating a 'context' for handling the request. Mobile agents inhabit contexts, and it is through the membrane of the context that an agent communicates with the agent host.

An instance of the type that defines the service method is created within the context. Although everything within a context really constitutes an agent, it is the instance of the particular type that the programmer identifies as 'the agent'. This is because typically any other objects are mostly merely data, created and manipulated by the instance of the 'agent type'.

After the agent is created, it runs. By virtue of the separation the context provides, the agent is free to move away from its current host while the method is executing. The act of stopping a running agent, moving it to a new host, and restarting it requires process migration. Any reasonable mobile agent environment will support process migration.

The host waits for the agent to return and publish a result. When this happens, the result is communicated back to the client via the Web service stub. This process is illustrated in Figure 6.

The operation of the Web service stub and agent host act to deliver a method invocation to a mobile agent. The programmer can take the abstract view that SOAP requests are handled directly by a mobile agent.

After create the mobile agents and mapping web services on those, use the Multi agent Encryption Protocol which presented in section 2, then sent these multi agents to destination nodes and in destination node messages will be decrypted and used by that node.

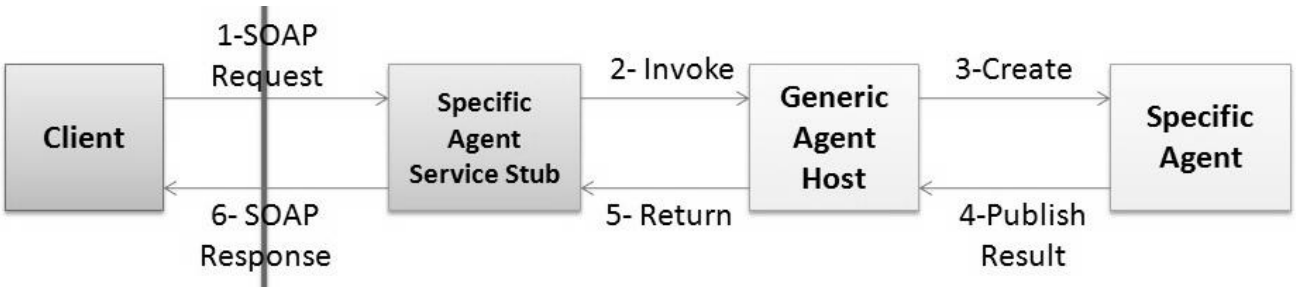

Fig 6: Request mapping on Mobile Agent

In this protocol, we use AES encryption algorithm for cryptography, but can use any other algorithm instead this. Also in my approach using two level cryptography and multi agent for transfer split message, but if required security is lower, you can use one level cryptography and few mobile agents, beside if need very high level of security, can use several level cryptography and more than mobile agents.

In next section, we evaluate our approach and descript the resulted advantages of this.

5. EVALUATION

Our approach against the older ways, have several advantages that descript those here:

- Increasing security(by using multi agent encryption protocol).
- Reduces bandwidth.
- Possibility use of different cryptography algorithm in our protocol.
- Possibility local accessing to host data and do computing on there.
- Possibility use this approach in parallel and distributed algorithms
- Moderates the effects of high latency,(since there are only agent send- and return-trips)
- Is robust in intermittently connected networks.

Above detail, obtained by compare with other ways to implement of web services and proposal approach to establish the security for those.

We also evaluated the proposed algorithm according to the message crack probability. It is supposed that P is the crack probability for one mobile agent. According to the proposed approach, all the agents should be cracked to encrypt the message. Therefore, for n mobile agents we have:

$$P_m = P^n \quad (5)$$

Where P_m is the probability for cracking the total message. It should be noted that the above probability is independent from the number of malicious hosts. The number of existing malicious host may only decrease the required time for the message cracking. figure 7 illustrates the above relation.

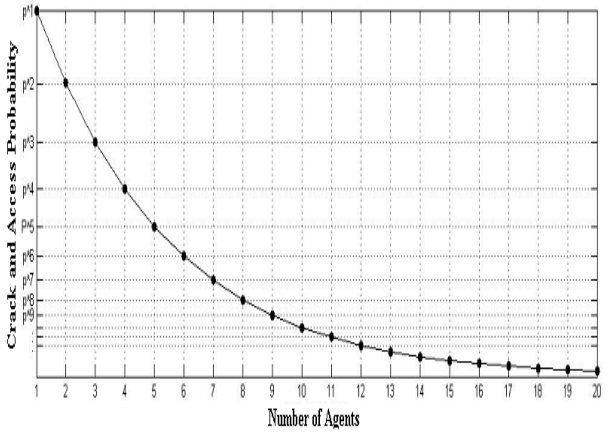

Fig 7: Message crack probability regarding number of agents

6. CONCLUSION

According to the open nature of networked environments and security challenge of such systems, We proposed a software-only, new approach for implementation of web services by using multi agents and using a multi agent encryption protocol to increasing security of web services in novel network and create efficient and flexible web services.

In our proposed approach mobile agents are used for implementing Web services. Clients can readily consume mobile agent functionality as though it was an ordinary Web service. When a mobile agent wants to consume a service, it is possible to transparently invoke either a legacy or mobile-agent backed Web service by doing dynamically-bound method invocations. A set of legacy Web services can be gradually replaced by a mobile agent-based implementation, which may be more amenable to optimization.

7. REFERENCES

[1] Abolfazl Esfandi, Ali Movaghar Rahimabadi, "Mobile Agent Security in Multi agent Environments Using a Multi agent-Multi key Approach", in Proc. 2nd IEEE International Conference on Computer Science and Information Technology, Vol. 4, August 2009, pp. 438-442.

[2] National Institute of Standards and Technology, "Announcing the ADVANCED ENCRYPTION STANDARD (AES)," Federal Information Processing Standards Publication, no. 197, Nov. 2001.

[3] Rosenthal Joachim, "A Polynomial Description of the Rijndael Advanced Encryption Standard", Journal of Algebra and Its Applications, Vol. 2(2), 2003, pp. 223-236.

[4] Xu Ke, "Mobile Agent Security Through Multi-Agent Cryptographic Protocols", PhD Thesis, Department of Computer Science and Engineering, University of North Texas, May 2004.

[5] Java Remote Method Invocation (RMI) Specification. 2001,Sun Microsystems, Inc.

[6] T. Erl, "SOA: Principles of Service Design, " Prentice Hall/Pearson PTR, 2007.

[7] E Nagy, B., "Deployment Descriptors" in Apache SOAP User's Guide . 2001.

[8] A. Singhal, T. Winograd and K. Scarfone, "Guide to Secure Web Services, " National Institute of Standards and Technology Special Publication, 2007.

[9] Common Language Infrastructure (CLI), Partition I: Architecture. 2002, ECMA International, Geneva.

[10] S. Chollet and P. Lalanda, "An Extensible Abstract Service Orchestration Framework," IEEE International Conference on Web Services (ICWS), 2009.

[11] J. G. R. Sathiaseelan, S. A. Rabara and J. R. Martin, "Multi-Level Secure Framework for Composite Web Services," ACM International Conference Proceedings (ICIS), pp. 580–585, 2009.

[12] A. Ginige and S. Murugesan, "The Essence of Web Engineering – Managing the Diversity and Complexity of Web Application Development," IEEE Multimedia, vol. 8, no.2, pp. 22-25, Apr.–Jun.2001.

[13] G. H. Hwang, Y. H. Chang and T. K. Chang,"An Operational Model and Language Support for Securing Web Services," IEEE International Conference on Web Services (ICWS), 2007

[14] Yildiz, B., Fox G., and S. Pallickara, "An Orchestration for Distributed Web Service Handlers" International Conference on Internet and Web Applications and Services ICIW 2008, June 8-13, 2008 - Athens, Greece

[15] A. Menezes, P. Van. Oorschot and S. Vanstone, "Handbook of Applied Cryptography," CRC Press, October 1996 – 5th reprinting, Aug. 2001, ch 12.

[16] Jana, D., Chaudhuri, A. and Bhaumik, B. 2009 Privacy and Anonymity Protection in Computational Grid Services. International Journal of Computer Science and Applications, Vol, 6, No, 1, pp. 98-107.