Framework on Hybrid Network Management System Using a Secure Mobile Agent Protocol

Olutayo B. Ajayi, Adebayo. D. Akinde, and A. T. Akinwale
Department of Computer Science, University of Agriculture, Abeokuta, Nigeria

tayoajayi@unaab.edu.ng; ada2@chams.com; atakinwale@yahoo.com

Abstract

As configuration of network services is faced with wide-spread deployment problems requiring considerable human efforts and involvement, Mobile Agent based Network Management System become a central concern. Moreover, the recent developments in the area of mobile agent based network management and ever improving Java Programming language have provided important tools for designing Secure Mobile Agent based Network Management Protocol (SMAN). Again, a roaming agent on a network consumes significant network bandwidth which implies that their frequency and number must be regulated. In a sensitive and intelligent network where the Agent behaviour can be altered dynamically during the lifetime, the proposed system must be genuinely secure with less bandwidth overhead. So, it is necessary to design packet filtering and secure protocol using the modified multi-signcryption protocol for the purpose of efficiency.

Keywords: Mobile Agent, Security, Protocol, Network Management, Aglet, SNMP

Introduction

The current network is characterized by its increasing distribution, its dynamic nature, and the complexity of its resources, due to the increasing requirement of different services (Yang, Galis, Mota, & Michalas, 2003). Network management essentially involves monitoring and controlling the devices connected in a network by collecting and analyzing data from the devices (Stallings, 1999).

The current trend is to deploy mobile agents to manage large heterogeneous networks. Mobile agents are special software objects that have the unique ability to transport itself from one system in a network to another in the same network (Feng, 2002).

One of the possible approaches is to automate the installation and configuration steps using a mobile-agent based Plug-and-Play (PnP) architecture for service configuration.

Related Works

As networks are growing and becoming more distributed, the need for better management through available distributed technologies is being realized. According to Kona and Xu (2002), mobile agent technology has long been pursued but its applications in network management are still rudimentary. Bieszczad,

Pagurek, and White (1998) described theoretical views on application of mobile agents for network management that lack concrete implementation. Gavalas, Greenwood, Ghanbari, and Mahogany (2000) presented the application of mobile agents in bulk transfer of network monitoring data, data aggregation and acquiring atomic SNMP table views. They analyzed the usage of mobile agents in network management with regard to the bandwidth utilization. The work addresses the issue of mobile agents for network monitoring, but did not consider provisioning services. Pinheiro, Pohylisher, & Caldwell (2000) described a conceptual model which collects management related data across a changing set of networked components and periodically compute aggregated statistics using mobile agents. More concentrated towards aggregation of network monitoring data and exploring mechanisms for agent adaptation.

SMAN Architecture

The proposed flexible architecture, Secure Mobile Agent based Network Management (SMAN) framework, is a hybrid model, which has features of secure mobile agent protocol as well as Simple Network Management Protocol (SNMP). The architecture forms a layer over the conventional SNMP based management that ensures the advantages of SNMP are not lost and also serves the purpose of managing legacy SNMP based systems. SMAN gives the manager the flexibility of using SNMP model or SMAN depending on the management activity that is involved. This architecture has many advantages over the existing architectures. Some of the advantages are stated below:

- The repetitive request/response handshake is eliminated
- Reduces design risk by allowing decisions about the location of the code pushed towards the end of the development effort
- Resolves problems created by intermitted or unreliable network connections
- Real time notifications
- Parallel executions (or load balancing) where large computations are divided amongst processing resources.
- Offers an alternative to or complementing SNMP security in network management system

In the proposed architecture the station assumes responsibilities of a client. All managed nodes are servers, which have mobile agent environment and respond to SNMP queries from mobile agents when they visit the context servers and manipulate data locally. When the client in the SMAN needs access to data in a network-connected device, it does not talk directly to the server over the network but dispatches a mobile agent to the server's machine. On arriving at the servers' machine, the mobile agent makes its request and return to the management station with the results. The architecture provides Java-compliant interfaces to network management services. Aglet Software Development Kit (ASDK) (http://www.trl.ibm.com/aglets/) is the agent development environment to be used because of its modular structure, easy-to-use API for programming of mobile agents and excellent documentation. To interact with the SNMP agent, we use AdventNet SNMP (http://www.adventnet.com/). It provides a set of Java tools for creating cross platform Java and Web-based SNMP network management applications. AdventNet provides a set of classes, which could be used to facilitate communication between managed device (a device with SNMP agent like Routers), and an SNMP manager or management application.

The SMAN architecture consists of the following major components:

- Management application (MAP)
- Mobile Agent Execution Environment (MAEE)
- Secure Mobile Agent Producer (SMAP)
- Mobile Agents (MA)
- Modified Multi-signcryption protocol (MMSP)

In the SMAN architecture, the mobile agents are provided with:

- The list of nodes to be managed
- SNMP statistics of interest
- Health functions (Lienwand, 1996) defined by the user

The mobile agent development environment is the Aglet Software Developer Kit (ASDK), which provides a modular structure, easy-to-use API for programming of mobile agents and excellent documentation.

Figure 1 and Figure 2 show the hybrid model of SMAN and architecture for network management using secure mobile agents. The administrator/manager is given the flexibility of deciding whether to use SNMPv3 or mobile agents. Every mobile agent enabled network device has to offer a mobile agent context server. The mobile agents hosted in the context servers communicate with the local SNMP agent via SNMP based management applications.

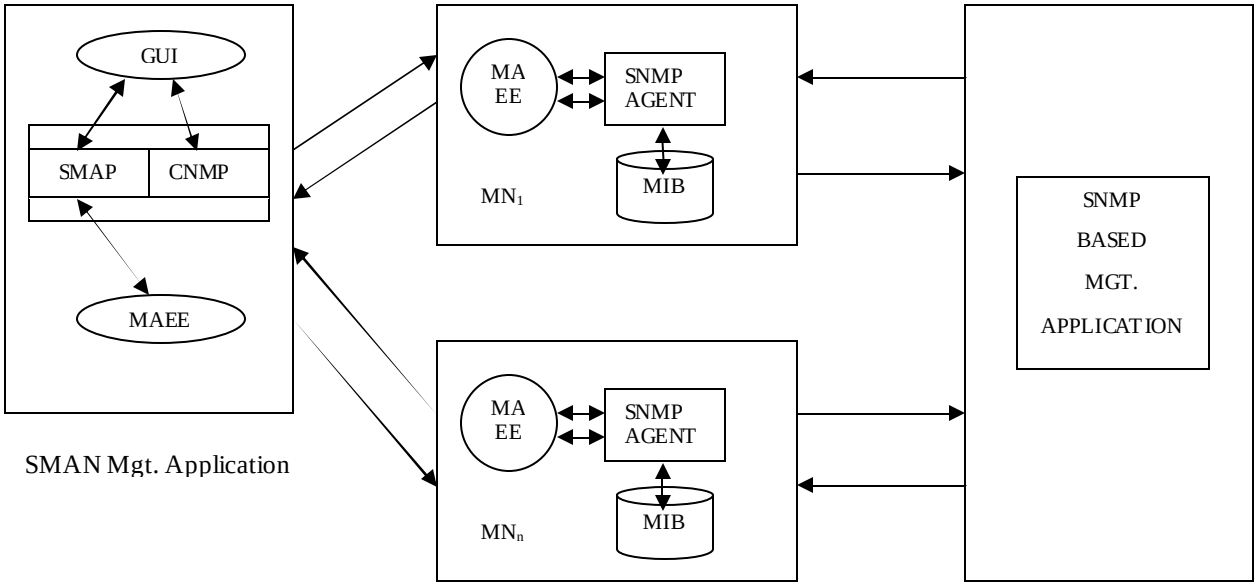

Figure 1: Hybrid SMAN Model

Keys:
GUI – Graphical user Interface
CNMP – Conventional Network Management Protocol
MIB – Management Information Base
MN_i – Managed Nodes (i.e. Network Devices) where i = 1 to n

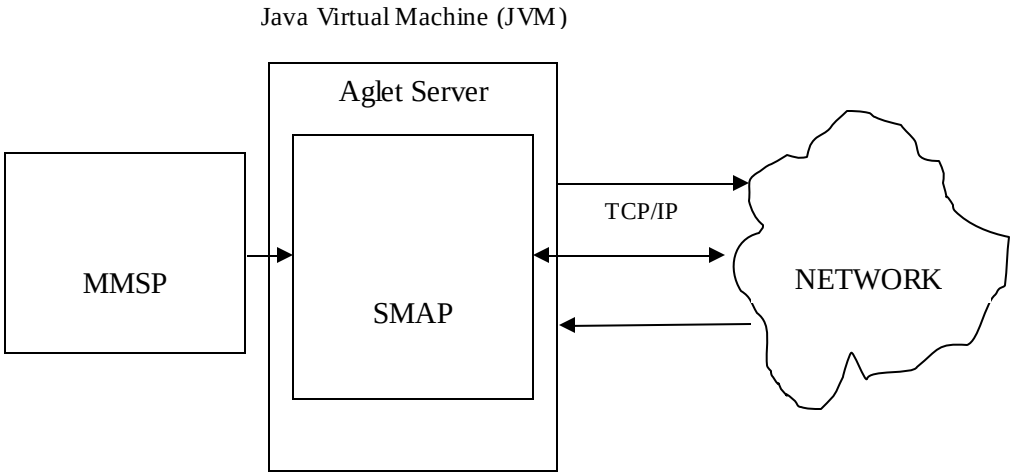

Figure 2: SMAN Station

The Aglet Server (Tahiti) runs on every network device as the context server for incoming mobile agents. The agents are subject to security policies that are contained in the Modified Multi-signcryption protocol (MMSP) designed in this work. The arriving agents are authenticated and there after communicate with SNMP agent via UDP packets. The advantage of this process is that no actual traffic is generated at all since the sockets are directed towards the 'loopback' device. At the end of the mobile agent task on the station, it dispatches itself to the next destination on its itinerary. Finally, the agent is disposed of at the end of its tasks.

An attacker may tamper with the agent (aglet) state and must be protected against an eavesdropping attack as it will contain sensitive administrative information. Hence, the agent data state are protected in order to provide authentication and confidentiality using the protocol described in subsequent sections.

Security Issues

Despite the attraction of mobile agent technology, security is still a major concern. Security is an even more important issue when the critical data is carried by a mobile agent (Lange & Oshima, 1999; Papastavrou, Samaras, & Pitoura, 2000). Indeed, while agents can be used to extract data for query purposes, the agents are prone to attack and hence the security of data in the agent is of prime concern (Pang, Catania, & Tan, 2003). One important issue is the malicious agent problem, where an agent that executes on a host attacks other agents or local resources. A second security concern is the malicious host problem (Yee, 1997). An agent is under complete control of its host, which may steal or modify agent information or even destroy the agent. The solution is to prevent the information from being disclosed to a host using robust secure protocol.

Most of the research work into security is concentrating on the malicious agent issue, by advancing techniques that isolate the execution of agent from the rest of the system. However isolating on its own is only a first step for security. A security framework for agent architecture must furnish further properties. It is important that agent that visits a trustworthy host must be able to authenticate the information that it furnishes. Again, a host that sends an agent out must possess ways to ensure that agent gets to their destinations unaltered (Bryce, 2000).

The fact that SNMP uses the unreliable, connectionless UDP rather than reliable, connection-oriented TCP reduces its security. An attacker can masquerade as a management station or a net-

work device and send out malicious UDP packets to the well-known SNMP ports (161, 162) or corrupting ongoing SNMP request-response sessions (Pashalidis & Fleury, 2002).

The core of our secure agent system builds a protocol that is called Multi-signcryption protocol that provides user authentication, integrity and confidentiality for the agent transactions and Agent Transfer Protocol (ATP) over the network. The multi-signcryption protocol is a cryptographic method that fulfills both the functions of secure encryption and digital multi-signature for multi-users, at a cost smaller than that required by multi-signature-then-encryption (Mitomi & Miyaji, 2001; Pang et. al., 2003; Seo & Lee, 2004).

Mitomi and Miyaji (2001) proposed a multi-signcryption protocol which combined a multi-signature with the encryption function. However, since their protocol can not provide message confidentiality, it cannot prevent a malicious attacker from obtaining the information in the messages. Pang et al. (2003) proposed a modified multi-signcryption protocol to achieve message confidentiality. However, since their protocol fixes the order of multi-signers beforehand, it does not satisfy the need for order flexibility. Moreover, it cannot provide non-repudiation. Seo and Lee (2004) analyzed the weaknesses of these previous multi-signcryption protocols and proposed a new multi-signcryption protocol. Their protocol provides not only message confidentiality, non-repudiation and order flexibility but also other requirements for secure and flexible multi-signcryption. It is believed to be more efficient. Therefore, in this work, we adapt modified Seo multi-signcryption protocol referred to in this work as MMSP and use it to design our secure mobile agent protocol.

Initialization and Notations

Let p, q be sufficient large primes with p = 2q + 1, and let $G \in Z_p^*$ have order q. Each managed node MN_0, MN_1, …., MN_n generates a pair of asymmetric key pairs (x_i, y_i), where $x_i \in Z_p^*$ and $y_i = g^{x_i} \bmod p$, and publishes the public key y_i along with its identity information ID_i through a Certificate Authority (CA). The MA itinerary (itireq corresponds to M_0) represents the original itinerary used to query or collect information from other managed nodes. Other notations used are stated below:

- MN_i : the i-th network gateway which belongs to the i-th managed node
- SMAN : the management center of an apartment complex
- NET : the network environment
- $E_{a,b}$: an elliptic curve over a finite field $GF(p^m)$, either with $p \geq 2^{150}$, m = 1 or p = 2, m ≥ 150 ($E_{a,b}: y^2 = x^3 + ax + b (p > 3)$, $E_{a,b}: y^2 + xy = x^3 + ax^2 + b (p = 2)$, $4a^3 + 27b^2 \neq 0 \pmod p$)
- q : a large prime number whose size is approximately of $|p^m|$
- G : a point with order q which is chosen randomly from the points on $E_{a,b}$
- $ENC_K(\cdot), DEC_K(\cdot)$: the encryption and decryption algorithms of a private key cipher system with the key K
- $H(\cdot)$, $hash(\cdot)$: a one-way hash function
- x_i : the secret key of the i-th manager who uses the MN_i, $x_i \in R [1, ..., q - 1]$
- Y_i : the public key of the i-th manager who uses the MN_i, $Y_i = x_i G$
- ‖ : denotes concatenation

Basic Solution

In this section, we present a basic solution for secure network management services by applying an EC based signature protocol to SNMP based Context Servers (CS). We append the EC-DSS (Elliptic Curve based Digital Standard Signature) scheme (Menezes, Ooorschot, & Vanstone, 1997) to the existing Network Management System (NMS) for user authentication and integrity of data. We assume that the existing NMS already establishes a common secret key K_i between MN_i and the Aglet (Tahiti) server of the Managed Nodes, and provides confidentiality through a private key cipher algorithm with K_i. Our basic solution is as follows.

[**EC-DSS Generation and Encryption phase**]

1. MN_i generates a signature on the itinerary data M_i as follows:
    a. MN_i chooses random $k_i \in R [1, ..., q - 1]$, and computes $r_i = k_i G \pmod q$
    b. MN_i computes $s_i = (H(M_i) + r_i x_i) \cdot k_i^{-1} \pmod q$
2. MN_i encrypts M_i with K_i, i.e., it generates $C_i = ENC_{Ki} (M_i)$.
3. MN_i sends (r_i, s_i, C_i, ID_i) to the SMAN.

[**EC-DSS Verification and Decryption phase**]

1. After the CS receives (r_1, s_1, C_1, ID_1), (r_2, s_2, C_2, ID_2), ...,(r_n, s_n, C_n, ID_n) from network gateways, it decrypts the C_i and obtains the itinerary data M_i of MN_i.
2. CS verifies the signature (r_i, s_i) of MN_i as follows:

    (a) CS computes $r_i' = (H(M_i)G + r_i Y_i) \cdot s_i^{-1} \pmod q$.

    (b) CS checks $r_i = r_i'$.

SMAN Protocol Using EC Multi-signcryption

In this section, we used a secure mobile agent protocol for network management services in network environments. Our protocol consists of four procedures such as registration procedure, mobile agent creation procedure, mobile agent execution procedure, and mobile agent arrival procedure. It provides confidentiality and integrity for the itinerary data, and user authentication using EC Multi-Signcryption. An overview of the proposed security model of the SMAN protocol is shown in Figure 3.

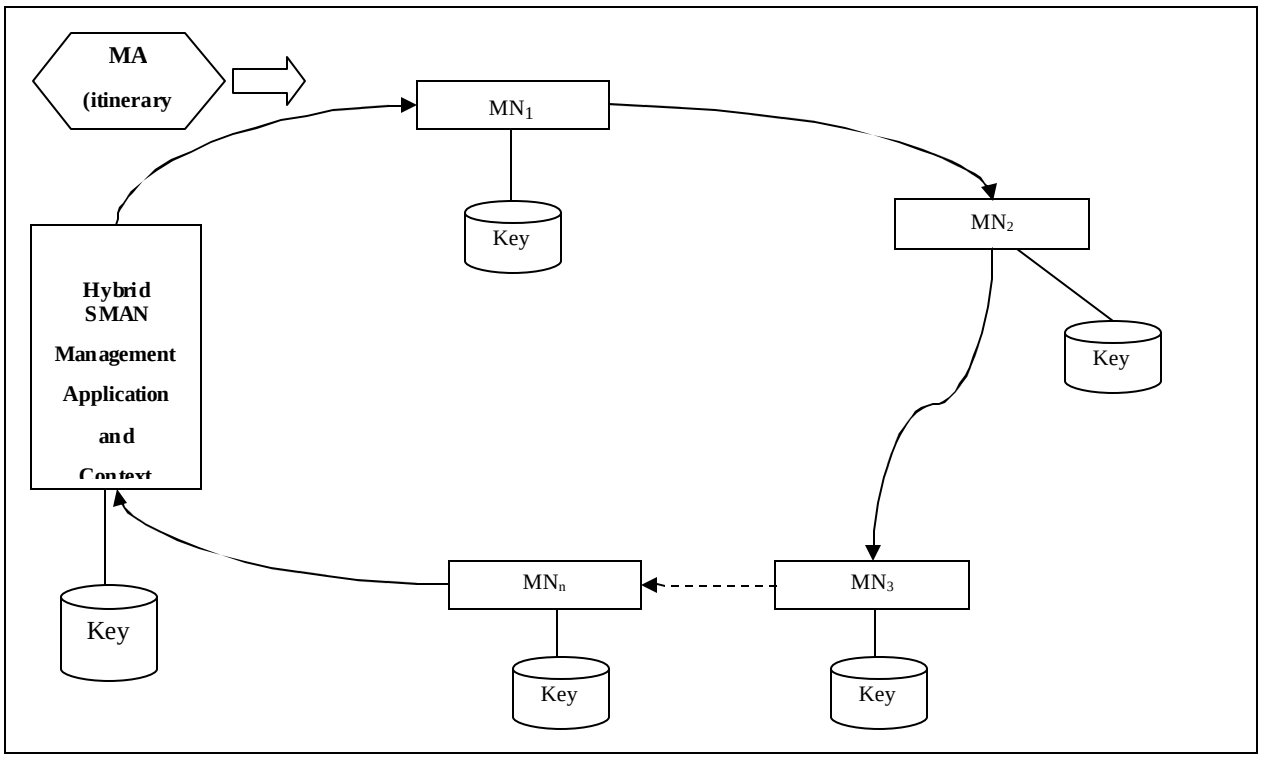

Figure 3: Security Model for SMAN Protocol

Certification Procedure

In this procedure, each manager $U_i (1 \leq i \leq n)$ registers his own public key and address at the management center, SMAN.

1. U_i gives his public key certificate and address information to the SMAN.
2. After the SMAN checks U_i's identity and address, it stores U_i's identity ID_i, public key Y_i, address, and MN_i information in the database of the CA (Certification Authority).

Preparation and Creation Procedure

In this procedure, the SMAN calls a mobile agent MA and determines the migration path of MA, $MA_{route} = MN_1||MN_2||...||MN_n$. Then it creates itinerary request message itireq, and generates a signature on itireq as follows:

1. SMAN chooses random number $k_C \in R [1, ..., q - 1]$ and computes $R_C = k_C G$.
2. SMAN computes $r_C = H(itireq||ID_C||R_C) \pmod q$ and $s_C = (x_C + r_C) \cdot k_{-1}^{C} \pmod q$.

SMAN gives itireq, MA_{route}, and signature, (ID_C, r_C, s_C) to the MA, and the MA migrates to the first manager's network gateway, MN_1 with them.

Execution Procedure

1. After the MA has migrated to $MN_i (1 \leq i \leq n)$, MN_i checks the itireq and MA_{route}.
2. MN_i verifies the SMAN's signature and generates the EC Multi-Signcryption on its itinerary data, M_i as follows:

[**Verification phase of the SMAN's signature**]

(a) MN_i computes $R'_C = s_C^{-1} \cdot (Y_C + r_C G) = s_C^{-1} \cdot (x_C + r_C)G = k_C G$.

(b) MN_i checks whether $H(itireq||ID_C||R'_C) \pmod q = r_C$, or not. If the equation holds, then it performs the following EC Multi-Signcryption phase. Otherwise, it reports the failure to the SMAN.

[**EC Multi-Signcryption phase**]

(a) MN_i chooses $k_i \in R$ [1, ..., q − 1], and computes a session key $K_i = hash(k_i \cdot Y_C) = hash(k_i \cdot x_C G)$ by using the SMAN's public key and k_i.

(b) MN_i computes the signature $r_i = H(M_i||ID_i||K_i) + r_{i-1} \pmod q$ and $s_i = (x_i + r_i) \cdot k_i^{-1} \pmod q$ by using received $r_{i-1} (1 \le i \le n, r_0 = r_C)$ from MA. And, it generates $C_i = ENC_{Ki}(ID_i||M_i)$ by encrypting (ID_i, M_i) with K_i. The EC Multi-Signcryption message is composed of the multi-signature (r_i, s_i) and the cipher text C_i. (r_i, s_i) are for user authentication and the integrity of M_i, and C_i is for the confidentiality of M_i.

3. MN_i gives the EC Multi-Signcryption message (ID_i, r_i, s_i, C_i) to the MA. Here, $r_i (1 \le i \le n)$ is connected to r_{i-1}. So, if the SMAN knows only r_n of the last signer, MN_n, then it can compute r_i of the previous signers, $MN_i (1 \le i \le n-1)$. Therefore, the MA removes r_{i-1} from (ID_1, s_1, C_1), ..., $(ID_{i-2}, s_{i-2}, C_{i-2})$, $(ID_{i-1}, r_{i-1}, s_{i-1}, C_{i-1})$, and it stores (ID_i, r_i, s_i, C_i).

4. If i = n, then MA migrates from the MN_n to the SMAN. Otherwise, the MA migrates from the MN_i to MN_{i+1}.

Arrival Procedure

After the MA finishes the travels of the migration path MA_{route}, it arrives at the SMAN.

1. MA gives (ID_1, s_1, C_1), ..., $(ID_{n-1}, s_{n-1}, C_{n-1})$, and (ID_n, r_n, s_n, C_n) to the SMAN.

2. SMAN performs the following EC Multi-UnSigncryption to verify and decrypt the EC Multi-Signcryption message.

[**EC Multi-UnSigncryption phase**]

(a) For i = n, ..., 3, 2, 1, SMAN computes the session key K'_i using its private key x_C, MN_i's public key Y_i, and (r_i, s_i).

i. SMAN computes $u_i = x_C \cdot s_i^{-1} \pmod q$ and $K'_i = hash(u_i \cdot r_i G + u_i Y_i) = hash((r_i + x_i) \cdot u_i G) = hash(x_C k_i G)$.

If $K'_i = K_i$, then the SMAN can decrypt C_i. And it can obtain the itinerary data M_i and ID_i of the MN_i.

ii. SMAN computes $r_{i-1} = r_i - H(M_i||ID_i||K'_i) \pmod q$. If the signature, r_{i-1}, is recovered then the SMAN lets i = i − 1 and performs steps i and ii again.

(b) If the verification is finished correctly then the SMAN can confirm its own signature, $r_C (= r_0)$.

3. If the EC Multi-UnSigncryption phase is performed successfully and all itinerary data M_1, ...,M_n of MN_1, ..,MN_n are decrypted, then the SMAN stores M_1, ...,M_n.

4. SMAN terminates the MA's execution.

Analysis of the SMAN Protocol

In this section, we analyze the security of our mobile agent protocol according to the security requirements of message confidentiality, message integrity, user authentication, non-repudiation,

and robustness. Then we analyze the efficiency of our protocol in comparison with the basic solution.

Security Analysis

1. **Message Confidentiality**: Message confidentiality means that it is computationally infeasible for a malicious attacker to gain any partial information on the content of the EC Multi-Signcryption message. In our protocol, if an attacker intercepts the mobile agent, MA, and searches the data in MA, then he can obtain the EC Multi-Signcryption messages (ID_1, s_1, C_1), (ID_2, s_2, C_2), ..., (ID_n, s_n, r_n, C_n) of the itinerary data M_1,M_2, ...,M_n. And the attacker can compute $s_i^{-1} \cdot (r_i \cdot G + Y_i) = k_iG (1 \le i \le n)$ from the EC Multi-Signcryption messages. But, since the attacker cannot know SMAN's private key, x_C, he cannot compute session keys due to the difficulty of the elliptic curve discrete logarithm problem (Menezes et al., 1997). Therefore, it is computationally infeasible for the attacker to gain any information of the itinerary data, M_1, M_2, ..., M_n. Our protocol provides confidentiality for the itinerary data.

2. **Message Integrity**: Message integrity means that the communicated EC Multi-Signcryption messages cannot be manipulated by unauthorized attackers without being detected. Assume that a malicious attacker modifies MN_i's itinerary data and tries to forge MN_i's ($1 \le i \le n$) EC Multi-Signcryption message, (ID_i, r_i, s_i, C_i). The attacker can create the forged itinerary data M_i' by modifying M_i of MN_i. And then, he chooses $k_i' \varepsilon_R [1, ..., q-1]$ and can compute the session key $K_i' = hash(k_i' \cdot Y_C) = hash(k_i' \cdot x_C G)$ by using the SMAN's public key and k_i'. Moreover, the attacker can use the r_{i-1} by eavesdropping on the MA, and he can generate signature $r_i' = H(M_i' || ID_i || K_i') + r_{i-1} \pmod q$. But, since the attacker cannot know the U_i's (managers') private key x_i, he cannot compute $s_i' = (x_i + r_i') \cdot k_i'^{-1} \pmod q$. Even if he chooses a random x_i' and computes $s_i'' = (x_i' + r_i') \cdot k_i'^{-1} \pmod q$, the SMAN can verify that s_i'' is forged signature in the EC Multi-UnSigncryption phase. Therefore, the attacker cannot modify the itinerary data and cannot forge the EC Multi-Signcryption message. So, our protocol provides integrity for the itinerary data.

3. **User Authentication**: User authentication means the process whereby one party is assured of the identity of the second party involved in a protocol, and of whether the second party has actually participated. In our protocol, the SMAN can confirm the identity of the Administrator, U_i, through the ID_i included in the EC Multi-Signcryption message. In the EC Multi-UnSigncryption phase, the SMAN can assure that U_i actually participated. So, our protocol provides user authentication.

4. **Non-repudiation**: Non-repudiation means that neither Administrators nor the SMAN can falsely deny later the fact that he generated an EC Multi-Signcryption message. In our protocol, non-repudiation is provided as follows. Since each EC Multi-Signcryption message includes the administrator Ui's ($1 \le i \le n$) private key, x_i, anyone who does not know x_i cannot generate an EC Multi-Signcryption message instead of U_i. Therefore, if MN_i of U_i generates the EC Multi-Signcryption, he cannot falsely deny later the fact that he generated it.

5. **Robustness**: Robustness means that if the signature verification on a message fails, then it prevents such unauthentic messages from damaging a receiver. In our protocol, after the SMAN receives the EC Multi-Signcryption message from the MA, if the verification of $K_i' = hash(x_C \cdot s_i^{-1} \cdot r_iG + x_C \cdot s_i^{-1} \cdot Y_i) = hash(x_C k_i G)$ fails, then the SMAN cannot compute the session key, K_i. So, since it cannot decrypt the cipher text C_i, it can prevent damage by an unauthentic message or malicious code in the MA. Therefore, our protocol provides robustness.

Efficiency Analysis

We evaluate our protocol from a point of view of network and communication overhead, and compare our protocol with the basic solution. We use the number of point multiple and modular

multiplication to measure the computational cost, and the communicated message size to measure the communication overhead.

For convenience, we assume the following conditions:

(1) we denote the number of managed node gateways by n and the message size by |M| bits;

(2) the size of q is set to 160 bits;

(3) the output size of the cryptographic hash functions is 160 bits.

In the basic solution, since all MN_is transmit EC Multi-Signcryption messages $(ID_i, r_i, s_i, C_i)(1 \leq i \leq n)$ to the Aglet (Tahiti) Server of the SMAN at the same time, a network bottleneck can be happened. The total communication overhead of the basic solution is $n \cdot |M| + n \cdot |q| + n \cdot |H(.)| = n \cdot (|M| + 320)$. But, in our protocol, the total EC Multi-Signcryption messages from MN_1 to MN_n are (ID_1, s_1, C_1), ..., $(ID_{n-1}, s_{n-1}, C_{n-1})$, (ID_n, r_n, s_n, C_n), and the communication overhead is $n \cdot |M| + (n + 1) \cdot |q| = n \cdot (|M| + 160) + 160$. So, when compared with the basic solution, our protocol reduces the communication overhead to, at most, 50%. The amount of EC Multi-Signcryption messages to be stored in the Aglet (Tahiti) Server can also be reduced to, at most, 50%. Moreover, since the MA migrates autonomously and transfers EC Multi-Signcryption messages either between MN_i and MN_{i+1} or between MN_i and the Aglet (Tahiti) Server, the total remote interaction and network traffic can be reduced between them.

In the network overhead cost of our protocol and the basic solution, the point multiple is 1 for $MN_i(1 \leq i \leq n)$ and 2n for the Aglet (Tahiti) Server. In the case of 160-bit modular multiplication, our protocol is 1 for $MN_i(1 \leq i \leq n)$ and 2n for the Aglet (Tahiti) Server, but the basic solution is 2 for $MN_i(1 \leq i \leq n)$ and n for the Aglet (Tahiti) Server.

We have, so far, assumed that the same secret key K_i established previously between the $MN_i(1 \leq i \leq n)$ and the Aglet (Tahiti) Server in the basic solution, and evaluated the efficiency of the basic solution without computational and communication costs for key establishment. However, key establishment is complex; it results in heavy network and communication overhead. If the secret key is fixed in the basic solution, “key freshness” cannot be provided. If the basic solution simply refreshes the secret key periodically, then it can provide “key freshness.” But it has another security problem, i.e. it cannot provide “forward secrecy” or “backward secrecy”, and it is not secure against “known-key attack” (Menezes, 1997). Therefore, if we add a key establishment phase to the basic solution for overcoming these security problems, then the computational cost and communication overhead of the basic solution increase, and the efficiency decreases.

Unlike the basic solution, our protocol does not need a key establishment phase. So, our protocol is more efficient than the basic solution.

Scalability

We compared two different solutions for sending itinerary data on managed elements to test network overhead imposed by the SMAN. SMAN is compared to the centralized SNMP using AdventNet SNMP. The topology used on this experiment consists of one management station and three managed nodes (colleges: colnas.unaab.edu.ng; colanim.unaab.edu.ng; colerm.unaab.edu.ng) interconnected through a 100Mbps Ethernet LAN. All machines run Windows or Linux. The daemon snmpd, which is included in the Linux, is an SNMP agent that responds to SNMP request packets.

In order to evaluate the performance, we alternately repeat the elements using the itinerary {colnas, colerm, colanim, colnas, etc.}. The SMAN approach fetches the SNMP table and does some filtering based on the user’s requirement. The SMNP is implemented using AdventNet SNMP package. The manager sends SNMP UDP packets to a SNMP agent that responds to the manager.

The manager sends requests to all elements to be managed; one after the other. Thus, a new request is started after receiving the response from the previous one, until the last node receives a request and sends the response to the manager.

The response time of SMAN is measured as the mean time of the MA launching time and returning time. The centralized SNMP approach is measured as the mean time of the first GET message was sent out and the last result fetched back.

Table 1 lists the testing result:

Table 1: Response Time of SNMP and SMAN

	Centralized SNMP Approach	**SMAN**
1 host	0.69 Seconds	0.71 Seconds
2 hosts	0.9 Seconds	0.95 Seconds
4 hosts	1.2 Seconds	1.24 Seconds
30 hosts	4.9 Seconds	4.89 Seconds

From Table 1, the SNMP is a bit less when the hosts are small in performing the tasks. This is due to the fact that the SMAN is built on better architecture for handling mobility.

Regarding the health function computation, the SMAN daemon agent transfer less number of messages comparing to the SNMP method as shown in Table 2. Thus, the total message size is reduced and the bandwidth is saved.

Table 2: Communication Overhead of SNMP and SMAN Daemon Agent

	SNMP		**SMAN Daemon Agent**	
	No. of Messages	**Total Message Size**	**No. of Messages**	**Total Message Size**
Interface utilization	4	364Byte	1	35Byte
Interface Accuracy	3	275Byte	1	34Byte
IP Discard Rate	5	458Byte	1	37Byte

Conclusion

This work has presented a framework to design a hybrid model based on secure mobile agent protocol and SNMP strategies. The work gives network administrators flexibility of using any of the two approaches to exploit mobile agent technology in network management. The results show that as the managed nodes increases, the proposed techniques perform better than centralized approach. On this note, this paper has demonstrated that it is possible to develop a secure mobile agent network management system using Java components and cryptography. To this end, the paper has presented reasonable detail on design level view.

References

Bieszczad, A., Pagurek, B., & White, T. (1998). Mobile agents for network management. *IEEE Communication Survey,1*(1).

Bryce, C. (2000). A security framework for a mobile agent system. *Proceedings of the 6th European Symposium on Research in Computer Security, Toulouse, France*.

Feng, X. (2002). *Design and analysis of mobile agent communication protocols*. Master of Science (M.Sc) thesis, Institute of Computer Software, Nanjing University, China

Gavalas, D., Greenwood, D., Ghanbari, M. & Mahogany, M. O. (2000). Advance network monitoring applications based on mobile intelligent agent technology. *Computer Communications*.

Kona, M. K., & Xu, C. (2002). *A framework for network management using mobile agents*. Retrieved from http://www.ernie.eng.wayne.edu

Lange, D., & Oshima, M. (1999). Mobile agents with Java: The Aglet API. In D. Milojicic, F. Douglis, & R. Wheeler (Eds.), *Mobility: Process, computers, and agents* (pp. 495-512). Reading, Massachusetts, USA: Addison-Wesley Press.

Menezes, A. J., Ooorschot, P. C., & Vanstone, S. A. (1997). *Handbook of applied cryptography*. CRC.

Mitomi, S. & Miyaji, A. (2001). A general model of multi-signature schemes with message flexibility, order flexibility, and order verifiability. *IEICE Transaction on Fundamentals, E84-A*(10), 2488-2499.

Pang, X., Catania, B., & Tan, K. (2003). Securing your data in agent-based P2P Systems. *Proceedings of 8th International Conference on Database Systems for Advanced Applications (DASFAA '03)*.

Papastavrou, S., Samaras, G., & Pitoura, E. (2000). Mobile agents for World Wide Web distributed database access. *IEEE Transactions on Knowledge and Data Engineering, 12*(5), 802-820.

Pashalidis, A., & Fleury, M. (2002). *Secure network management within an open-source mobile agent framework*. Department of Electronic Systems Engineering, University of Essex, UK.

Pinheiro, R., Pohylisher, A., & Caldwell, H. (2000). *Mobile agents for aggregation of network management data*. Telecordia Technologies, University of Warwick, Bell South Telecommunications.

Seo, S. & Lee, S. (2004). Secure and flexible multi-signcryption scheme. *Proceedings of ICCSA 2004, Lecture Notes in Computer Science 3046* (pgs 689-697). Springer-Verlag.

Stallings, W. (1999). *SNMP, SNMPv2 and RMON: Practical network management*. Addison-Wesley.

Yang, K., Galis, A., Mota T., & Michalas, A. (2003). *Mobile agents security facility for safe configuration of IP networks*. EU IST project MANTMP, funded by commission of EU.

Yee, B. S. (1997). *A sanctuary for mobile agents*. Technical Report CS97-537, UC San Diego, Department of Computer Science and Engineering.

Biographies

Olutayo B. Ajayi received his B.Sc. and M.Sc. degrees in Computer Science. Currently, he is concluding his PhD. in Computer Science. He is Chief System Programmer at the ICT Resource Centre, University of Agriculture, Abeokuta, Nigeria. He is also a part-time lecturer in the Department of Computer Sciences at the same University. His research interests include mobile agents, network security and web applications.

Prof. A. D. Akinde is a renowned scientist and researcher in the area of mobile agents and information science. He recently retired as a professor at Obafemi Awolowo University, Ile-ife, Nigeria in the Department of Computer Science.

Dr. A. T. Akinwale is currently the head of department, Computer Science Department at the University of Agriculture, Abeokuta, Nigeria.