Secure Mobile Agents on Ad Hoc Wireless Networks

Evan Sultanik[*], **Donovan Artz,**[*] **Gustave Anderson,**[†] **Moshe Kam,**[†] **William Regli,**[*]
Max Peysakhov,[*] **Jonathan Sevy,**[*] **Nadya Belov,**[*] **Nicholas Morizio,**[*] **Andrew Mroczkowski**[*]
Department of Computer Science[*] Department of Electrical and Computer Engineering[†]
College of Engineering
Drexel University
Philadelphia, PA 19104-2875

Abstract

This paper describes **SWAT**, a Secure Wireless Agent Testbed. Our goal is to create an integrated environment to study information assurance for mobile agent systems on ad hoc wireless networks. The present SWAT consists of dozens of mobile hosts, both PDAs and laptops, and hundreds of both static and mobile software agents. In deploying the testbed, we have developed novel mechanisms for integrating autonomous agent technologies with public-key and symmetric key encryption to support secure communication, at multiple OSI layers, among groups of hosts and agents. The paper describes the architectural technology used in SWAT, the integration challenges, as well as applications for group collaboration, network health monitoring and system security at both the agent and the host level.

Introduction

The Secure Wireless Agent Testbed (SWAT), under development at Drexel University, is a live laboratory to study integration, networking and information assurance for next-generation wireless mobile agent systems. Specifically, the SWAT infrastructure, as conceptualized in Figure 1, consists of PDA-based computing platforms (mostly HP iPAQs) on an 802.11b wireless network with ad hoc routing. The security framework uses a combination of symmetric and public-key cryptography to support encrypted communication at both the network and the agent application layers. A novel feature of SWAT is the ability to support secure group communication, via shared key generation, for groups and sub-groups of computing hosts and agents. Security is monitored by agents that manage keys, assess network traffic patters and analyze host behaviors. Using this framework, agents can revoke access rights for suspicious hosts or agents and adaptively re-route traffic at the network layer to improve the information integrity of the overall system. Lastly, agents provide the implementation framework for a number decentralized user applications, including those for user authentication, collaboration, messaging and remote sensor monitoring.

Key Challenges. Building SWAT required we address many typical obstacles that exist for transition of artificial intelligence systems (in this case, mobile agent systems) into real-world use. This paper discusses two main challenges:

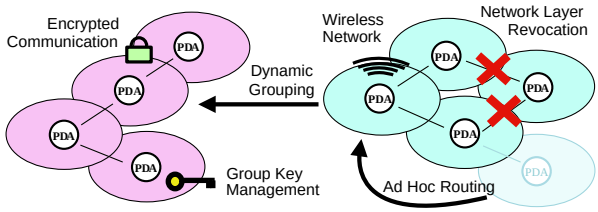


Figure 1: Conceptual view of SWAT and its features.

- **Mobile agents on ad hoc networks:** Inherent with the notion of an ad hoc network is the idea that nodes are able to be dynamically added to and deleted from the network. This is directly contradictory with the "static network topology" assumption that most agent infrastructures make. Various research efforts (Caripe *et al.* 1998) have studied subsets of these problems, including those working on dynamic service discovery (e.g., the CoABS Grid (Kahn & Cicalese 2002)), but we are far from having a comprehensive set of solutions.

  SWAT has developed a number of techniques for agents to reason about and act on network-layer information. Agents are able to modify the network state; make decisions about their itineraries (i.e., if they are mobile agents) based on network topology; and adapt their communication modalities to avoid network congestion. For example, when agents communicate, the connectivity of the agents may not mirror the connectivity of the network—hence, a broadcast message from one agent may result in massive quantities of network traffic. One SWAT innovation addresses the difficult problem of implementing general multicast communication on ad hoc networks by allowing agents select how to multicast based on current network topology.

- **Security for mobile agents:** Security for mobile agent systems has previously been addressed in a number of ways (Subrahmanian *et al.* 2000; Gray *et al.* 1998; Gray 2000): as a theoretical problem (i.e., a formal representation of trust, "How do I know if I can trust agent *x*?"); as a software problem (i.e., "How can I tell if the code for this agent has been tampered with?"); and, most recently, as a problem in economics and decision theory (i.e., trusted bidders, antisocial agents, etc.).

SWAT addresses the need for a mobile agent information assurance framework that includes cryptography and the ability for different groups of agents to generate secure communication channels within the overall agent community. Further, agents must be able to reason about security groups and communications in a manner that allows them to adapt to a dynamic security environment in which hosts may become compromised, networks may get attacked and malicious agents may need to be identified and contained.

Security engineering for mobile agent systems is a vast problem. SWAT includes a set of agent security capabilities that are critical for improving the state of information assurance for agent networks. First, SWAT contains agents that use the cryptographic infrastructure to perform tasks such as authentication, group key generation, revocation, non-repudiation, and threat profiling/detection. These actions can be performed at the user, host or agent level.

Second, SWAT hosts have agents meta-reason (i.e., reason about their own activities) using both security and network information. For example, what if a host in the network becomes compromised but simple removal of the host would result in a network discontinuity? In this case, agents analyze network topology and decide, from an information assurance standpoint, if its better to kick the compromised host out of the SWAT, reroute messages around it or just ignore it as it can do only limited harm.

Architectural Technology

A *mobile agent* is a self-contained program that can dynamically traverse a network on its own accord, possibly acting on behalf of a user or another party (Pham & Karmouch 1998). In an *ad hoc network*, any host in the network may serve as a router for any other host in the network (Lee, Su, & Gerla 2001). In this way, the host in an ad hoc network can dynamically fulfill the role of a router, thus enabling it to adapt in real time to changes in network topology—leading to network dynamism that can significantly influences the behavior of an agent system. Lastly, critical in any modern computing or information system is the need for security and information assurance.

While there is a vast community studying agent technologies, little work exists on to effectively integrate mobile agents, ad hoc networking and practical information assurance technologies. For example, while mobile agents have been used for discovering routes in an ad hoc networks (Bandyopadhyay & Paul 1999), little has been published regarding how agent systems can be designed to positively exploit the features of an ad hoc network. Agent security research has usually addressed how agents representation and reasoning about concepts such as "trust" among agents, or agents' beliefs, desires and intentions (Subrahmanian *et al.* 2000). Little exists on how to integrate mobile agents with a cryptographic infrastructure and how to adapt centralized security techniques to peer-to-peer networks.

This section briefly reviews SWAT's architectural technologies, touching on the state-of-the-art in mobile agent systems, ad hoc networking and security mechanisms.

Agent Frameworks

Requirements for the SWAT agent framework include:

- provide true agent mobility;
- be proven in production (i.e., tested, scalable, supported);
- be suitable for handheld computing;

There are currently many different approaches to and implementations of mobile agent architectures (Cohen *et al.* 1997; Gray *et al.* 2001; Hoile *et al.* 2002; Jennings, Sycara, & Wooldridge 1998; Sadeh *et al.* 1999; Sycara *et al.* 2001). SWAT employs the **The Extendable Mobile Agent Architecture** (EMAA) from Lockheed Martin's Advanced Technology Laboratories[1]. The EMAA framework includes autonomous and asynchronous agents as well as management mechanisms for agent mobility, agent events and inter-agent communication (Lentini *et al.* 1998). Architecturally, EMAA is composed of three core layers: *Docks*, *Servers*, and *Agents*. *Docks* provide the execution environment on a host in which all other components are executed. *Servers* provide "heavy-weight" or fixed services while *Agents* are "lighter" and mobile, each having task lists and itineraries. While highly autonomous, EMAA agents can still receive and respond to commands from a controlling authority, thus balancing between totally autonomous behavior and cooperation to command centers that may be controlled by human users (McCormick *et al.* 1999).

In addition to agent messaging and agent migration, EMAA also supports event messaging. The **Distributed Event Messaging System** (DEMS) is a key part of EMAA responsible for all event-based communications within the framework. EMAA utilizes DEMS because traditional systems like CORBA and RMI use static bindings to locate and communicate with agents, and since EMAA's agents are mobile their location changes from time to time (McCormick *et al.* 1999). DEMS was created to follow the Java Event model; event messages themselves are delivered by *Transmission Agents*.

Ad Hoc Networks

SWAT is implemented for an ad hoc network environment, in which hosts have the ability to dynamically identify routes and forward packets between hosts that are not within direct wireless range of each other. The properties ad hoc routing protocols have been detailed in the IETF Mobile Area Networking working group's RFC 2501 (Macker & Corson 1998). An example of the ad hoc network topology for a set of mobile and wireless hosts is shown in Figure 2.

SWAT employs **Ad-Hoc On-Demand Distance Vector Routing** (AODV), an on-demand "destination-routing" protocol. AODV implicitly uses hop counts recorded during the route-discovery process as distance vectors for selection of the best route to a destination. As with other on-demand routing protocols, AODV defines mechanisms for two central routing activities: route discovery, finding paths to destinations; and route maintenance, repairing paths with broken links (Perkins & Royer 1999).

[1] `http://www.atl.lmco.com`

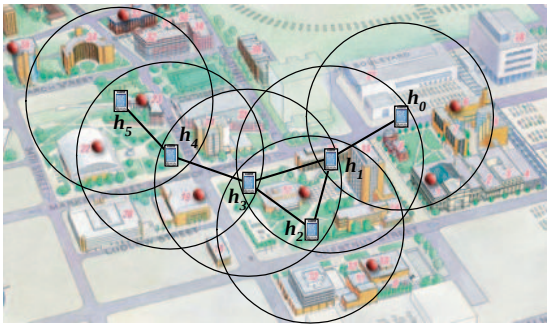


Figure 2: A wireless, ad hoc network of PDAs. The wireless range of each PDA determines direct connections.

Agent systems occasionally require multi-cast or broadcast mechanisms to communicate with other agents. This poses special challenges to ad hoc networks. In a wired network, broadcast traffic is generally sent out on all interfaces except that on which it came in; this, together with loop-freedom in the network, eliminates the problem of duplicates being passed around the network forever. The nature of wireless ad hoc networks requires each node to maintain a record of those broadcast packets which it has already seen for the purpose of duplicate elimination. Different protocols deal with this problem in different ways. For example, with the AODV protocol, it has been proposed that nodes use unique identifiers to track messages (Perkins, Belding-Royer, & Das 2001). Multicast deals with communication between a group of hosts, and is often used by audio messaging applications. Currently, a true multicast architecture has been proposed only for the AODV and OLSR protocols (Royer & Perkins 2000; Jacquet & et al. 2001); however, no non-simulation implementation is presently available for either protocol.

Security Protocols and Mechanisms

As part of SWAT, we examined how to use security mechanisms for traditional (non-agent) applications and fixed networks. Objectives included:

1. Avoid a single point of failure by using decentralized and distributed services;
2. Practice forward security, thus precluding the use of previous cryptographic secrets by a potential attacker;
3. Employ controls to ensure the integrity of data;
4. Consider reactive security for the cases where preventative security fails; and,
5. Deploy a discretionary access control system to enforce resource privileges.

The established security technologies integrated into SWAT include:

- **CLIQUES** (Steiner, Tsudik, & Waidner 1998; Amir *et al.* 2000) is a protocol for key generation and management. Among other realizations, it provides implementations of Group Diffe-Hellman (GDH) and Tree Group Diffe-Hellman (TGDH) algorithms (Amir *et al.* 2002).
- **Spread** is a client/server application for reliable group communication (Amir & Stanton 1998). The server is distributed (Spread Server segment) and each client (Secure Spread client) is logged onto the local server segment.
- **Secure Spread** is an implementation of the CLIQUES protocol using Spread as its communication platform.
- **Semi-Trusted Mediator** (SEM) is an algorithm for user revocation (Boneh *et al.* 2001). In SWAT, when a user is revoked, the SEM prevents it from participating in symmetric key generation.
- **IPSec** is used to en/decryption of traffic on the network layer, using symmetric key(s) generated by Secure Spread (Kent & Atkinson 1997).

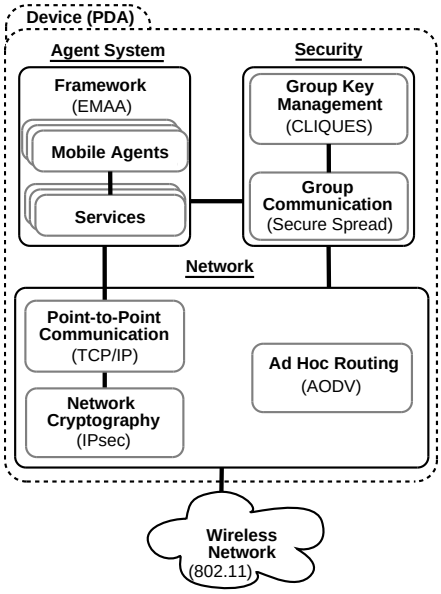


Figure 3: SWAT Host Architecture. Each SWAT host integrates technologies for Agents, Networking, and Security.

SWAT System Integration

Each host in the SWAT is composed of several of the above technologies which integrate the agent system, the network, and security infrastructure. Figure 3 shows the components of a SWAT host, and how these components are connected. The EMAA agent framework contains both mobile agents, and static agents (services). The security components of a host include group key management, implemented by Secure Spread (using TGDH), and group membership revocation, enforced by the integration of SEM and Secure Spread. The agent framework is connected to the security components, enabling an agent (or the whole agent system) to join or leave a group, with the permission to join controlled by the SEM. The network components enable secure point-to-point communication for the agent framework, as well as reliable group communication for the security components. Point-to-point communication is implemented using standard TCP/IP and is secured using IPSec. All network communication is routed through SWAT's 802.11b wireless network using the AODV routing protocol.

When two or more hosts are present in SWAT, connections are made between hosts through SWAT's wireless net-

OSI-RM Layer	SWAT Technologies
Application	EMAA, Spread, Agent Applications, Security Applications
Transport	TCP (EMAA), UDP (Spread)
Network	AODV, IPSEC, Netfilter
Data Link	WEP, MAC Address Filtering
Physical	802.11b, 802.11a, Bluetooth, USB, RS-232

Table 1: SWAT technologies used across the OSI-RM.

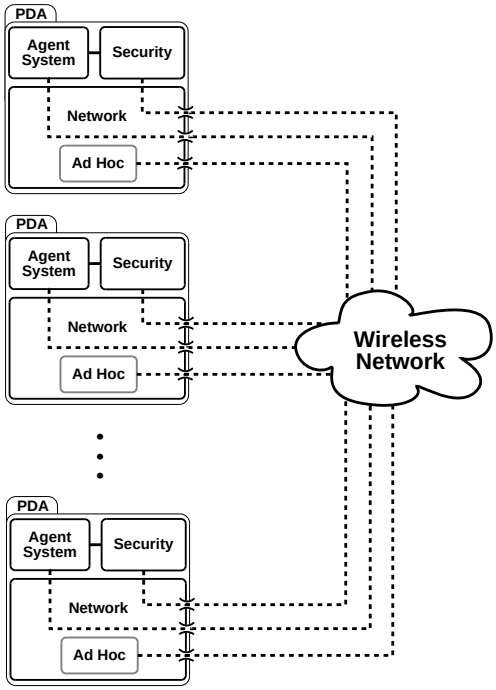


Figure 4: SWAT Network Architecture. The SWAT network, containing any number of hosts, comprises Agent System, Security, and Ad Hoc Routing traffic.

work. The SWAT network is characterized by three types of traffic: *agent system*, *security*, and *ad hoc network management and routing*. *Agent system* traffic is generated by EMAA, and includes agent messaging, agent migration, and event messaging. *Security traffic*, which is generated by Secure Spread and SEM, includes the generation of group keys, the entry or exit of a group member, and the revocation of group membership. *Ad hoc network management traffic*, generated by AODV routing, includes the maintenance and discovery of network routes in SWAT. The types of traffic in SWAT and the ability of each host to send and receive traffic of each type are illustrated in Figure 4. SWAT employs technologies at each layer of the OSI-RM (Zimmerman 1980) to provide a comprehensive network foundation. The specific technologies used by SWAT are listed in Table 1.

SWAT's security mechanisms enable both hosts and agents to belong to one or more *groups*. *Groups* provide encrypted channels for private communication. Using CLIQUES, a group of hosts or agents can securely generate a shared symmetric key. Using this key, group members may encrypt messages that can only be decrypted by group

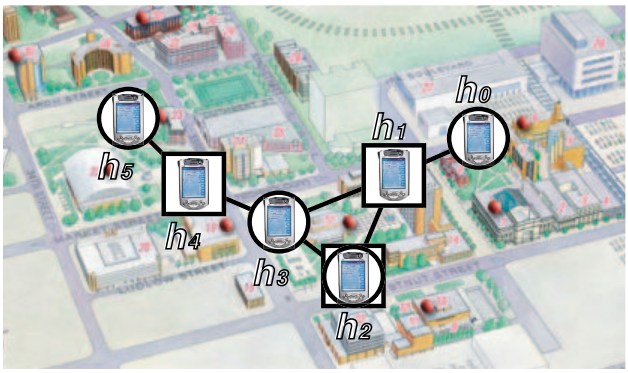


Figure 5: SWAT Groups. SWAT agents and hosts can form groups for secure communication. In this example, each host is a member a subset of two groups, denoted by shape.

members. If a group member is revoked from a specific group, the SEM will prevent that member from participating in the next CLIQUES generated key for that group. A specific host or agent may have have membership in multiple groups simultaneously. Figure 5 illustrates an example of SWAT using 6 hosts, where each host is in one of 3 groups (each represented with a different shape). Noting the example's topology and the group formed by hosts h_1 and h_4, encrypted group messages between h_1 and h_4 must pass through a non-member host, h_3. However, host h_3 is not able to decrypt group messages outside of its group membership with hosts h_0 and h_5. Hence, it is possible for members of SWAT to perform secure group communication, even when group messages pass through non-member hosts.

In integrating these technologies, SWAT has begun to address many issues that future systems engineers of ad hoc, wireless agent systems will also need to consider:

1. **Integration of host and agent system**, providing an interface between agents and the local computing host;
2. **Integration of host and security**, protecting each host against active and passive network attack;
3. **Integration of agent system and network**, enabling agents to reason about and adapt to SWAT's dynamic network; and
4. **Integration of agent system and security**, so agents can form and communicate in secure groups, identify compromised agents and hosts, and make networking decisions to improve overall information assurance.

We discuss these issues in the subsequent sub-sections.

Integration of Host and Agent System. Agent mobility depends on the capability of a host to transfer an agent's runtime state. EMAA, being implemented in Java, uses Java's *thread serialization* capability to implement agent mobility. SWAT uses a pre-release Java virtual machine (JVM) for ARM processors found in HP iPAQs[2] which provides this feature. Using this JVM, each iPAQ in SWAT is capable of sending and receiving mobile agents.

SWAT's agent messaging is dependent on the continuous connectivity of the underlying hosts. This connectivity is achieved through use of the AODV wireless, ad hoc routing protocol. SWAT iPaq hosts use a modified version of the AODV Linux kernel module provided by NIST.[3] Significant activity within the SWAT team is directed toward enhancing this implementation of AODV to ensure that the intended functionality is preserved as more features are added to SWAT. Multicast AODV (MOADV) provides a mechanism for a group of agents located on a group of hosts to perform group messaging in an efficient manner. MOADV is a feature currently under development in SWAT.

Integration of Host and Security. SWAT precludes passive attacks (i.e., eavesdropping) by encrypting *all* network traffic between hosts, at multiple levels,[4] using CLIQUES to provide forward security against "man in the middle" attacks during key generation. The encryption is performed using software provided by the OpenSSL Project.[5] Encryption and decryption at each network interface is performed using an implementation of IPSec.[6] SWAT Agents, even when performing plain text agent messaging, can assume that their communication is only readable within the agent system.

Because each host in SWAT is networked it is vulnerable to network attack. For this reason, each SWAT host uses a firewall to proactively filter and record any unexpected network transmissions. The firewall is implemented using a feature now integrated into the Linux kernel: netfilter.[7] Netfilter enables each host to control the incoming, outgoing, or locally processed traffic with fine granularity.

The SWAT uses a customized Linux kernel and software distribution. The Linux kernel is based on the work of R. King,[8] and uses the Familiar Linux distribution.[9] The SWAT team has evolved part of Familiar's general purpose distribution into a robust, secure operating system by removing many unnecessary applications and services to minimize possible security holes, and to optimize the space requirements given each iPAQ's limited amount of storage. However, the greatest degree of customization lies within the kernel. Many configuration options are modified to accommodate the variety of extra peripherals (network cards, GPS, etc.) that are utilized by the SWAT iPAQs.

[2] `http://www.blackdown.org/`

[3] `http://w3.antd.nist.gov/wctg/aodv_kernel/`

[4] SWAT does not rely on the weak encryption provided by WEP, which is easily broken, but creates a multi-layered, multi-group VPN over 802.11b using IPSec.

[5] `http://www.openssl.org`

[6] `http://www.freeswan.org/`

[7] `http://www.netfilter.org/`

[8] `http://www.arm.linux.org.uk/`

[9] `http://familiar.handhelds.org/`

Integration of Agent System and Network. There are two qualities of ad hoc, wireless networks that affect how agents can communicate. *Route redundancy* is the existence of more than one possible route between two hosts in a network. As SWAT's network topology changes, route redundancy between any two hosts may also change. Hence, at any time, an agent may have a choice as to how its message is sent over SWAT's network. The second quality affecting agents is the continuously changing topology in a wireless, ad hoc network. If a host becomes undesirable or unreachable, an agent may wish to avoid that host. Mobile agents use an *itinerary* to determine which hosts they will migrate to in the course of completing tasks.

SWAT introduces the idea of *network meta-reasoning agents* that can exploit route redundancy and enable the hosts, docks, services and other agents to adapt to changing network states. For example, if a host becomes compromised, or if a host is removed for security purposes, SWAT's network meta-reasoning agents can modify their itineraries or re-route agent messages to increase robustness and reduce the effect of compromised hosts. Given a disconnected or compromised host, each agent either creates a new itinerary or reports failure to its originating host.

The dynamic nature of ad hoc networks creates constant gains and/or losses of services and agents caused by network merges and disconnections. As two separate networks are combined, it may be necessary to eliminate duplicate services and agents. As a network is separated, it may be necessary to compensate for the loss of services and agents in each of the resulting networks. SWAT is currently studying how to best address these issues separately for each applications.

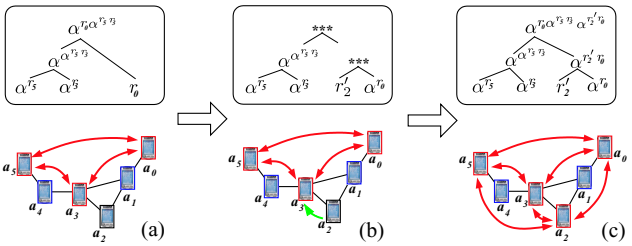


Figure 6: Agent a_2 joins the red group: (a) a_2 not in red group; (b) a_2 asks a_3 to join the red group; (c) a_3 sponsors a_2 and a new group key is generated.

Integration of Agent System and Security. SWAT agent security is realized through secure group communication. To generate a shared group key, CLIQUES relies on the TGDH key agreement protocol. Rather than collecting the key contribution of each member in sequence, requiring linear time, TGDH imposes a binary tree structure over the members of a group, reducing key generation to logarithmic time. Moreover, the tree structure enables CLIQUES to efficiently recompute a key in the event that a member joins or leaves

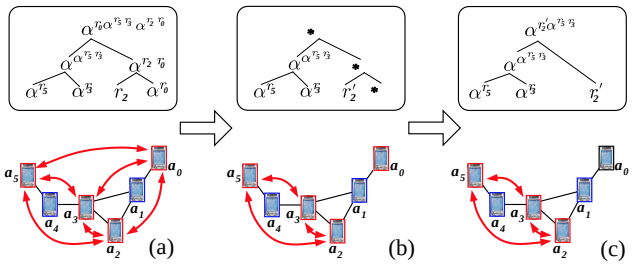


Figure 7: Agents a_2, a_3 and a_5 revoke access of a_0; (a) a_0 is a member of red group; (b) a_2 abandons a_0's key contribution; (c) red group generates a new key without a_0.

(or is revoked from) a group. Figure 6 shows the progress of CLIQUES in generating a new shared key for a group admitting an new member. Figure 7 shows the progress of CLIQUES in generating a new shared key, given that the group has revoked one of its members. The logical tree associated with each topology graph in these figures shows how the contributions of each agent are combined to form a new shared group key.

The SEM records each revocation, and prevents revoked members from participating in the next key generation. If we were to enforce *instantaneous* revocation, every communication would have to be routed through the SEM. Due to the high computational and network cost of this approach, the present SWAT enforces revocation only at the time of the next key generation.

Traditional security mechanisms have been applied to agent systems, including trail obscuring, code obfuscation, and audit logging (Greenberg *et al.* 1998). Beyond traditional security mechanisms, agents provide new possibilities in computer security. SWAT advances a new approach to agent-based security that allows agents themselves to reason about security and the how they affect it. In SWAT, agents can implement their own active security measures.

Example SWAT Applications

We present three agent-based applications that currently are living within the SWAT.

An Agent-based Whiteboard. Collaboration among SWAT users is done via a secure multicast, multi-group, whiteboard. This whiteboard utilizes a peer-to-peer architecture that employs EMAA agents to deliver messages to members of each group. Multiple users are allowed to work collaboratively without losing any of the features provided by a traditional centralized whiteboard. Whiteboard communications can be described as a complete graph in which each node represents a server on a host which uses agent messages to communicate with all the other participants. Because the whiteboard is decentralized, there is no central server node, or even a server present in the whiteboard architecture. Each of the clients reside on a different host and are responsible for performing services that would otherwise be performed by a server. All of the changes produced by each of the nodes are relayed to all the other nodes through communication channels that are available in each host. As can be seen in Figure 8, the whiteboard's agents will only deliver annotations to members of the group the agent itself belongs to. For example, in (a), the annotation will not be delivered to the whiteboard in (c), because (c) is not a member of group **A**. However, the annotation in (c) will not appear in (a), as the whiteboard in (a) is not a member of group **B**. Whiteboard (b), which is a member of both **A** and **B**, sees both annotations.

Network Topology and Resource Monitoring. The Network Topology and Resource Monitor (Figure 9) utilizes agents that traverse the network sampling information about each host. This sampling includes information about the state of hosts equipped with remote sensing equipment, such as cameras and GPS receivers. The topology is visually represented as an undirected graph in which each node is a host and the edges denote zero-hop routes. For example, if two hosts are within physical wireless range of each other (and therefore their communications do not require routing), an edge will be drawn between them on the graph.

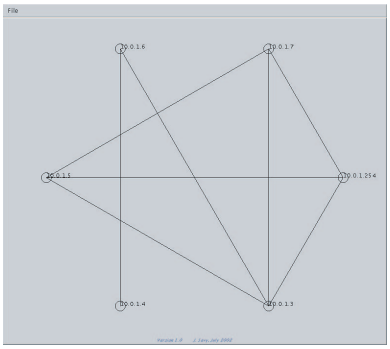
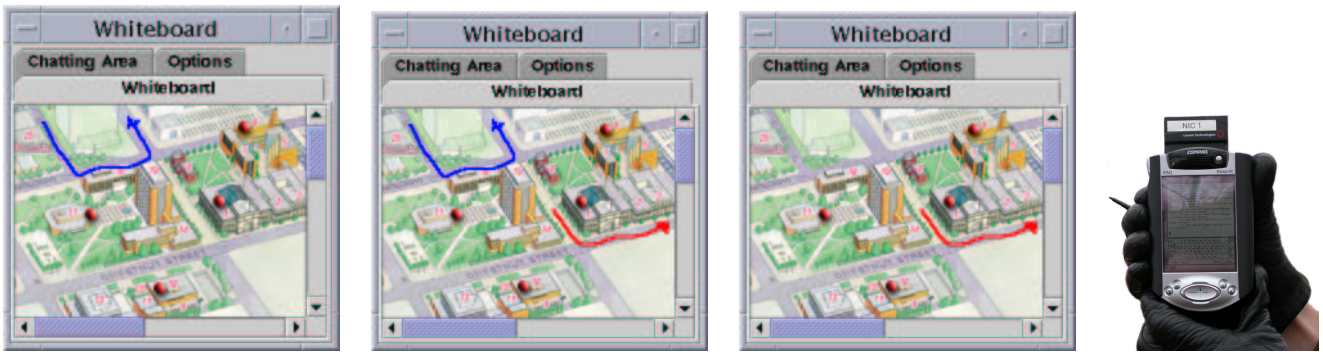


Figure 9: Network monitoring agents.

Resource monitoring is vital for diagnosing the state of the network and agent security. In SWAT, the resource monitoring agents report to *Judge* agents, which collect the resource usage data and attempt to identify suspicious hosts or traffic and report this to privileged users. In order to enable the *Judge* agents to make these decisions, SWAT imposes a strict structure upon the data collected by the resource monitoring agents. One technique SWAT uses is rule based, in which information obtained by the agents is used to generate rules which update a knowledge base maintained at the target *Judge* agent. In the present SWAT, *Judge* agents use a previously created static decision tree to determine the appropriate actions given the current state of the network (i.e., revocation of group membership, removal of a host from the network, etc). Future SWAT plans call for use of more sophisticated machine learning techniques to allow *Judge* agents to do host and agent profiling.

Meta-Reasoning for Information Assurance. Network meta-reasoning agents use a state description of SWAT's underlying ad hoc network to reason about how agents communicate. Based on this state description, a formal representa-



(a) Member of group **A** (b) Member of groups **A** & **B** (c) Member of group **B** (d) SWAT iPAQ

Figure 8: (a–c) Multi-group collaboration via the SWAT whiteboard; (d) a SWAT-enabled iPAQ.

tion of information assurance for agent messaging is used to evaluate the agent system's state with regard to a compromised host. For example, as *Judge* agents detect an intrusion, the network meta-reasoning agents determine what action (if any) should be taken against that intrusion. Possible actions include ignoring the compromised host, rerouting the network around the compromised host where possible, and removing the compromised host completely from the network. The utility of an action is found by by quantifying the effect of the compromised host on the integrity of messages among agents. Agents can determine, in real time, what action will provide the best balance of message integrity and routing efficiency.

Current SWAT Status

SWAT has been under development since March of 2002 by a team of about a dozen graduate and undergraduate students lead by two faculty investigators. The SWAT team has employed an iterative development process: first, bootstrap the technologies to get a baseline integration of agents, PDAs, networks and security; then, identify problems arising in the integration and find and implement candidate solutions; repeat. One major lesson learned in SWAT was that an integrated system consisting of many "moving" technologies cannot be designed neatly from top down and in advance. Many issues are found in real-time, when the system goes "live" and interaction among the components occurs. In the laboratory, the SWAT has remained fully operational for several days at a time. When in use, many new issues suddenly have to be identified and addressed (i.e., battery life, I/O rates with flash memory, network dropouts and beacon conflicts, adjusting routing in the Linux kernel, to name just a few). SWAT plans call for continued maintenance, refinement, stabilization and extension of the system through 2004. These activities include updates for new software and hardware systems, integration of new applications and collaboration with Lockheed Martin's ATL.

The current SWAT consists of two dozen mobile computing devices (HP iPAQs and Dell laptops), each hosting several dozen EMAA agents. Significant effort has gone into improving system robustness and stability, mostly through diagnosis of interoperability problems among the hardware and software layers of the system. SWAT's present use of 802.11b bandwidth indicates that the architecture can scale to double the current size (from 20-24 hosts to between 40-50 hosts) before migration to alternative wireless protocols becomes necessary (i.e., 802.11a or 802.11g). Creation of additional, network-intensive, mobile agent applications may precipitate this transition occurring sooner.

Present SWAT studies include an analysis of computational, memory, and bandwidth usage—in particular the performance of the key generation and distribution algorithms in the security framework. The limited computational power of the iPAQ nodes makes optimization of these techniques very important to ensure future scalability. We are also working toward complete elimination of the "single point of failure" problem by "agentizing" the SEM, which is presently located on a single host.

The most significant studies from the point of view of agent-based computing address how agents can effectively self-manage their own security infrastructure by making collective, yet decentralized, decisions. As noted earlier, networks meta-reasoning agents use utility-based model for assessing integrity of agent communications in the presence of a compromised host. With this model, agents alter the underlying network to improve their information assurance state. Other security agents are being developed to diagnose security problems based on network and host resource utilization profiles.

SWAT is currently being tested and validated in a number of practical scenarios. The main functional objective of SWAT is to provide users with tools for distributed, mobile, collaborative work and communication. There are many practical applications of such a system (i.e., police personnel at a sports event, medical personnel at an accident scene, emergency responders to a natural disaster, etc.). The SWAT working test scenario is a Drexel University "Campus Tour," (Figures 2 and 8) in which multiple tour guides equipped with SWAT-enabled PDAs provide multiple groups parents an introduction to the campus. As part of the ongoing process of integrative development, SWAT is currently being used and tested in two ways. First, informally, by the SWAT development team during group events, lunch outings, etc. Second, formally, by staging demonstrations and, eventually, fielding the units to users for a campus event.

Conclusions

Mobile agents, security, and ad hoc networks are emerging as essential components of next-generation computing and collaboration infrastructures. The Drexel SWAT is deploying an integrated system, using mostly PDA-based computing devices, in which to examine this convergence and the many new, interdisciplinary, research issues that it presents.

SWAT demonstrates how agents can be integrated with public-key and symmetric key encryption infrastructures to create multiple agent groups and facilitate secure inter-agent communication. Further, in building SWAT, we have identified and addressed several key research problems that cut across AI, wireless networks and security. SWAT introduces new ideas for how agents can reason with and manipulate their network topology to improve security and enhance application performance. With the testbed, we are examining how known security engineering mechanisms for attacker profiling and resource monitoring can be migrated to a decentralized, agent-based environment.

We anticipate continued emphasis on the combination of small wireless devices, agents, and information assurance in emerging applications. Future work for the SWAT team includes continued refinement and extension of these technologies—studying the integration issues associated with this convergence and identifying new problems for practitioners of agent-based computing.

References

Amir, Y., & Stanton, J. 1998. The spread wide area group communication system. Tech. Rpt. CNDS-98-4 Center for Net. & Dist. Sys., John Hopkins Univ.

Amir, Y.; Ateniese, G.; Hasse, D.; Kim, Y.; Nita-Rotaru, C.; Schlossnagle, T.; Schultz, J.; Stanton, J.; & Tsudik, G. Secure group communication in asychronous networks with failures: Integration and experiments. *ICDCS'00*.

Amir, Y.; Kim, Y.; Nita-Rotaru, C.; & Tsudik, G. On the performance of group key agreement protocols. *ICDCS'02*.

Bandyopadhyay, S., & Paul, K. Evaluating the performance of mobile agent-based message communication among mobile hosts in large ad hoc wireless network. *1999 ACM Wkshp on Modeling, Analysis & Simulation of Wireless & Mobile Systems*, 69–73.

Boneh, D.; Ding, X.; Tsudik, G.; & Wong, M. 2001. A method for fast revocation of public key certificates and security capabilities. *10th USENIX Sec. Symp.*, 297–308.

Caripe, W.; Cybenko, G.; Moizumi, K.; & Gray, R. 1998. Network awareness and mobile agent systems. *IEEE Comm.* 36(7):44–49.

Cohen, P. R.; Cheyer, A.; Wang, M.; & Baeg, S. C. 1997. An open agent architecture. Huhns, M. N., & Singh, M. P., eds., *Readings in Agents*. 197–204.

Gray, R. S.; Kotz, D.; Cybenko, G.; & Rus, D. 1998. D'Agents: Security in a multiple-language, mobile-agent system. Vigna, G., ed., *Mobile Agents & Security*, V 1419 *Lec. Notes in CS*. 154–187.

Gray, R. S.; Kotz, D.; Peterson, R. A.; Barton, J.; Chacon, D.; Gerken, P.; Hofmann, M.; Bradshaw, J.; Breedy, M. R.; Jeffers, R.; & Suri, N. 2001. Mobile-agent versus client/server performance: Scalability in an information-retrieval task. Picco, G. P., ed., *5th Int'l Conf. on Mobile Agents*, Lec. Notes in CS, 229–243.

Gray, R. S. Soldiers, agents and wireless networks. *2000 Conf. on the Prac. Ap. of Intel. Agents & Multi-Agents*.

Greenberg, M. S.; Byington, J. C.; Holding, T.; & Harper, D. G. 1998. Mobile agents and security. *IEEE Comm.* 36(7):76–85.

Hoile, C.; Wang, F.; Bonsma, E.; & Marrow, P. Core specifications and experiments in DIET *AAMAS'02*, 623–630.

Jacquet, P., et al. 2001. Multicast optimized link state routing. IETF Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-molsr-01.txt

Jennings, N. R.; Sycara, K.; & Wooldridge, M. 1998. A roadmap of agent research and development. *JAAMS* 1(1):7–38.

Kahn, M. L., & Cicalese, C. D. T. 2002. The CoABS grid. Goddard/JPL Wkshp on Radical Agent Concepts.

Kent, S., & Atkinson, R. 1997. Security architecture for the internet protocol. IETF Draft. ipsec-arch-sec-02.txt

Lee, S.-J.; Su, W.; & Gerla, M. 2001. Wireless ad hoc multicast routing with mobility prediction. *Mobile Networks & Ap.* 6(4):351–360.

Lentini, R. P.; Rao, G. P.; Thies, J. N.; & Kay, J. 1998. EMAA: an extendable mobile agent architecture. AAAI Wkshp on Software Tools for Dev. Agents.

Macker, J. P., & Corson, M. S. 1998. Mobile ad hoc networking and the IETF. *Mobile Comp. & Comm. Rvw.* 2(1):9–14.

McCormick, J.; Chacón, D.; Lentini, R.; McGrath, S.; & Stoneking, C. 2000. A distributed event messaging system for mobile agent communication. TR-01-02

Perkins, C., & Royer, E. 1999. Ad-hoc on-demand distance vector routing. *IEEE Wkshp on Mob. Comp. Sys.&Ap.*, 90–100.

Perkins, C. E.; Belding-Royer, E. M.; & Das, S. R. 2001. IP flooding in ad hoc mobile networks. IETF Draft. draft-ietf-manet-bcast-00.txt

Pham, V. A., & Karmouch, A. 1998. Mobile software agents: An overview. *IEEE Comm.* 36(7):26–37.

Royer, E. M., & Perkins, C. E. 2000. Multicast ad hoc on-demand distance vector (MAODV) routing. IETF Draft. draft-ietf-manet-maodv-00.txt

Sadeh, N.; Hildum, D.; Kjenstad, D.; & Tseng, A. 1999. MASCOT. Agents '99

Steiner, M.; Tsudik, G.; & Waidner, M. CLIQUES: A new approach to group key agreement. *ICDCS'98*, 380–387.

Subrahmanian, V.; Bonatti, P.; Dix, J.; Eiter, T.; Kraus, S.; Ozcan, F.; & Ross, R. 2000. *Heterogeneous Agent Systems*.

Sycara, K.; Paolucci, M.; van Velsen, M.; & Giampapa, J. 2001. The RETSINA MAS infrastructure. *JAAMS*.

Zimmerman, H. 1980. OSI reference model - the ISO model of architecture for open systems intercomm. *IEEE Trans. on Comm.* (425432).