# SAFE: Secure Agent roaming For E-Commerce

Sheng-Uei Guan* and Yang Yang

*Department of Electrical & Computer Engineering,*
*National University of Singapore,*
*10 Kent Ridge Crescent, Singapore 119260*
*Fax: (65) 779 1103*
*email:eleguans@nus.edu.sg*

* Corresponding Author

**Abstract**

The development of the Internet has made a powerful impact on the concept of commerce. E-commerce, a new way to conduct business, is gaining more and more popularity. Despite its rapid growth, there are limitations that hinder the expansion of e-commerce. The primary concern for most people when talking about on-line shopping is security. Due to the open nature of the Internet, personal financial details necessary for on-line shopping can be stolen if sufficient security mechanism is not put in place. How to provide the necessary assurance of security to consumers remains a question mark despite various past efforts. Another concern is the lack of intelligence. The Internet is an ocean of information depository. It is rich in content but lacks the necessary intelligent tools to help one locate the correct piece of information. Intelligent agent, a piece of software that can act on behalf of its owner intelligently, is designed to fill this gap. However, no matter how intelligent an agent is, if it remains on its owner's machine and does not have any roaming capability, its functionality is limited. With the roaming capability, more security concerns arise. In response to these concerns, SAFE, Secure roaming Agent For E-commerce, is designed to provide secure roaming capability to intelligent agents.

**Keywords:** agent transport, agent security, electronic commerce, mobile agents, roaming agents

## 1. Introduction

The introduction of the Internet is probably one of the most significant revolutions of the 20th century. With a simple click, one can connect to almost every corner of the world thousands of kilometers away. This presents a great opportunity for e-commerce. Despite its many advantages over traditional commerce, e-commerce has not taken off successfully. One of the main hindrances is security:

When it comes to online transactions, security becomes the primary concern. The Internet was developed without too much security in mind. Information flows from hubs to hubs before it reaches the destination. By simply tapping into wires or hubs, one can easily monitor all traffic transmitted. For example, when Alice uses her VISA credit card to purchase an album from Virtual CD Mall, the information about her card may be stolen if it is not carefully protected. This information may be used maliciously to make other online transactions, thus causing damage to both the card holder and the credit card company.

Besides concerns on security, current e-commerce lacks intelligence. The Internet is like the world's most complete library collections unsorted by any means. To make things worse, there is no competent librarian that can help readers locate the book wanted. Existing popular search engines are attempts to provide librarian assistance. However, as the collection of information is huge, none of the librarians are competent enough at the moment.

Intelligent agent is one solution to providing intelligence in e-commerce. But having an agent that is intelligent is insufficient. There are certain tasks that are unrealistic for agents to perform locally, especially those that require huge amount of information. Therefore, it is important to equip intelligent agents with roaming capability.

Unfortunately, with the introduction of roaming capability, more security issues arise. As the agent needs to move among external hosts to perform its tasks, the agent itself becomes a target of attack. The data collected by agents may be modified, the credit carried by agents may be stolen, and the mission statement on the agent may be changed. As a result, transport security is an immediate concern to agent roaming. SAFE transport protocol is designed to provide a secure roaming mechanism for intelligent agents. Here, both general and roaming-related security concerns are addressed carefully. Furthermore, several protocols are designed to address different requirements. An e-commerce application can choose the protocol that is most suitable based on its need.

### 1.1 Background on Agents

There has been a lot of research done on the area of intelligent agents. Some literatures only propose certain features of intelligent agents, some attempt to define a complete agent architecture. Unfortunately, there is no standardization in the various proposals, resulting in vastly different agent systems. Efforts are made to standardize some aspect of agent systems so that different systems can inter-operate with each other. In the area of knowledge representation and exchange, one of the most widely accepted standards is KQML (Finin, 1993) (Knowledge Query and Manipulation Language) developed as part of the *Knowledge Sharing Effort*. KQML is designed as a high level language for runtime exchange of information between heterogeneous systems. Unfortunately, KQML is designed with little security considerations because no security mechanism is built to address common security concerns, not to mention specific security concerns introduced by mobile agents. Agent systems using KQML will have to implement security mechanisms on top of KQML to protect themselves. In an attempt to equip KQML with 'built-in' security mechanisms, Secret Agent is proposed by (Thirunavukkarasu 1995).

Secret Agent defines a security layer on top of KQML. Applications will have to implement some special message format in order to make use of Secret Agent. Secret Agent has a number of shortcomings and is handicapped by the design of KQML.

Firstly, one requirement of Secret Agent is that every agent implementing the security algorithm must possess a key (master key). This master key is either a symmetric key or based on PKI. If the key is based on a symmetric key algorithm, it requires each agent to have a separate key with every other agent it corresponds with. If the agent intends to communicate with another agent that it has no common pre-established master key, a central authentication server is required to generate such a key. The problems introduced are key database management, authentication server protection and key transport/exchange security.

If the master key is based on PKI, the agent identity must be tightly tied with the key pair. This was insufficiently addressed in the design of Secret Agent, subjecting the algorithm to man-in-the-middle attack. In the SAFE transport protocol, agent identity and key pair are tightly integrated using digital certification.

Another prominent transportable agent system is Agent TCL developed at Dartmouth College (Gray 1997)(Kotz 1997). Agent TCL addresses most areas of agent transport by providing a complete suite of solutions. It is probably one of the most complete agent systems under research. Its security mechanism aims at protecting resources and the agent itself. Since some existing agent systems are already very strong in this area, Agent TCL 'seeks to confirm their sufficiency and either copy or redesign as appropriate' (Gray 1997). In terms of agent protection, the author acknowledges that 'it is clear that it is impossible to protect an agent from the machine on which the agent is executing… it is equally clear that it is impossible to protect an agent from a resource that willfully provides false information' (Gray 1997). As a result, the author 'seeks to implement a verification mechanism so that each machine can check whether an agent was modified unexpectedly after it left the home machine' (Gray 1997). In other words, it addresses agent integrity and provides certain level of traceability to the agents. The other areas of security, like non-repudiation, verification, identification, are not carefully addressed.

Compared with the various agent systems discussed above, SAFE is designed to address the special needs of e-commerce. The other mobile agent systems are either too general

or too specific to a particular application. By designing SAFE with e-commerce application concerns in mind, the architecture will be suitable for e-commerce applications. The most important concern is security as discussed in previous sections. Due to the nature of e-commerce, security becomes a prerequisite for any successful e-commerce application. Other concerns are mobility, efficiency, and interoperability. In addition, the design allows certain flexibility to cater to different application needs.

## 2. General Agent Transport

As a prerequisite, each SAFE entity must carry a digital certificate issued by SAFE Certificate Authority, or SCA. In this way, each agent, agent owner, and host will carry its own unique digital certificate. The certificate itself is used to establish the identity of a SAFE entity. Because the private key to the certificate has signing capability, this allows the certificate owner to authenticate itself to the SAFE community. An assumption is made that the agent private key can be protected by function hiding (Thomas 1998) (Other techniques are also discussed in the literature (Bem 2000) (Westhoff 2000) but will not be elaborated in this paper).

From the host's viewpoint, an agent is a piece of foreign code that executes locally. In order to prevent a malicious agent from abusing the host resources, the host should monitor the agent's usage of resources (e.g., computing resources, network resources). Agent receptionist will act as the middleman to facilitate and monitor agent communication with external party.

### 2.1 General Message Format

In SAFE, agent transport is achieved via a series of message exchanges. The format of a general message is as follow:

SAFE Message = Message Content + Timestamp + Sequence Number + MD(Message Content + Timestamp + Sequence Number) + Signature(MD)

The main body of a SAFE message comprises of message content, a timestamp and a sequence number. The message content is defined by individual messages.

A timestamp contains the issue and expiry time of the message. If the message arrives before the issue time of the message or after the expiry time of the message (assuming there is no time lag between the sender and receiver), the recipient should generate an alert to the message sender as well as the recipient's administrator. The default message lifetime (time duration between issue time and expiry time) is set in the SAFE community. However, individual entities can choose to set a different message lifetime based on their needs. The local setting will overwrite the general setting by SAFE. The general guideline is that the duration must be longer than the maximum tolerable time for message exchange to complete but less than maximum tolerable agent transport time.

To further prevent replay attack, message exchanges between entities during agent transport is labeled according to each transport session. A running sequence number is included into the message body whenever a new message is exchanged. In this way, if a message is lost during transmission or an additional message is received, the recipient will be able to detect it.

In order to protect the integrity of the main message body, a message digest is appended to the main message. The formula of the message digest is as follow:

Message Digest = MD5(SHA(message_body) + message_body)

The message digest alone is not sufficient to protect the integrity of a SAFE message. A malicious hacker can modify the message body and recalculate the value of message digest using the same formula and produce a seemingly valid message digest. To ensure the authenticity of the message, a digital signature on the message digest is generated for each SAFE message. In addition to ensuring message integrity, the signature serves as a proof for non-repudiation as well.

If the message content is sensitive, it can be encrypted using a symmetric key algorithm (e.g., Triple DES). SAFE does not provide a general key exchange protocol for general messages. The secret key used for encryption will have to be decided at a higher level.

To cater for different application concerns, three transport protocols are proposed: supervised agent transport, unsupervised agent transport, and bootstrap agent transport. These three protocols will be discussed in the following sections in details.

## 2.2 Supervised Agent Transport

Supervised agent transport is designed for applications that require close supervision of agents. Under this protocol, an agent has to request roaming permit from its owner or butler before roaming. The owner has the option to deny the roaming request and prevent its agent from roaming to undesirable hosts. Without agent owner playing an active role in the transport protocol, it is difficult to have tight control over agent roaming.

The procedure for supervised agent transport is shown in Figure 1.

### 2.2.1 Agent Receptionist

Agent receptionists are processes running at every host to facilitate agent transport. If an agent wishes to roam to a host, it should communicate with the agent receptionist at the destination host to complete the transport protocol. Every host will keep a pool of agent receptionists to service incoming agents. Whenever an agent roaming request arrives, an idle agent receptionist from the pool will be activated to entertain the request. In this way, a number of agents can be serviced concurrently. The number of agent receptionists in the pool should be set to the maximum number of acceptable concurrent visiting agents in the host. If the number of roaming requests exceeds the number of agent receptionists, the request will not be granted until some existing visiting agent leaves the host.

### 2.2.2 Request through source receptionist for entry permit

To initiate supervised agent transport, an agent needs to request for an entry permit from destination receptionist. Communication between visiting agent and foreign parties (other agents outside the host, agent owner etc) is done using an agent receptionist as a proxy. The request for entry permit is first sent to the source receptionist. The request contains the requesting agent's digital certificate and the destination's address. The source receptionist will forward the agent's digital certificate to the destination receptionist as specified in the agent's request.

The destination receptionist can inspect the requesting agent's information by reading its digital certificate and decide whether to issue entry permit based on its own authorization policy. If the request is granted, an entry permit is generated and returned to the requesting agent. The entry permit will contain a random challenge, a serial number, a validity period, the digital certificate of the requesting agent, and a digital signature by the destination receptionist on the entry permit.

The random challenge is used to authenticate the incoming agent. Its usage will be discussed later in the discussion of supervised agent transport protocol. For book keeping purpose, a serial number is included in the entry permit issued by a receptionist. This number should be unique to all entry permits issued by the same receptionist. A timestamp is also part of the entry permit. Different from timestamps on general messages, the timestamp on an entry permit specifies the validity of the entry permit. It is up to each receptionist to decide how long the issued entry permit remains valid. In order to prove the authenticity of the entry permit, the issuing receptionist needs to digitally sign the entry permit.

### 2.2.3 Request for roaming permit

Once the source receptionist receives the entry permit from the destination receptionist, it simply forwards it to the requesting agent. The next step is for the agent to receive a roaming permit from its owner/butler. The agent sends the entry permit and address of its owner/butler to the source receptionist. Without processing, the source receptionist forwards the entry permit to the address as specified in the agent request.

Agent owner/butler can decide whether the roaming permit should be issued based on its own criteria. If the agent owner/butler decides to issue the roaming permit, it will have to generate a session number, a random challenge, a freeze/unfreeze key pair. The roaming permit should contain the session number, random challenge, freeze key, timestamp, entry permit, and a signature on all the above from the agent owner/butler.

In order to verify that the agent has indeed reached the intended destination, a random challenge is generated into the roaming permit. A digital signature on this random challenge is required for the destination to prove its authenticity. This will be discussed in greater detail later.

For the issuing of every roaming permit, a key pair is generated. A public key is included in the roaming permit for agents to encrypt or freeze its sensitive code/data during roaming. When the agent reaches the destination, it can obtain the private key (unfreeze key) from its owner to activate itself.

Same as entry permit, roaming permit also contains a timestamp that specifies the validity of the permit. As a general guideline, the validity should be the same as that in the entry permit unless the validity specified in the entry permit is deemed inappropriate.

Since a roaming permit is issued based on the entry permit presented, the entry permit will be part of the roaming permit. In this way, a roaming permit issued to entry permit A can not be used as a valid roaming permit to enable an agent roaming using entry permit B.

Finally, to provide non-repudiation, the agent owner/butler will digitally sign the roaming permit.

### 2.2.4 Agent Freeze

With the roaming permit and entry permit, the agent is now able to request for roaming from the source receptionist. In order to protect the agent during its roaming, sensitive function and codes inside the agent 'body' will be frozen. This is achieved using the freeze key in the roaming permit. Even if the agent is intercepted during its transmission, the agent's capability is restricted. Not much harm can be done to the agent owner/butler. To ensure a smooth roaming operation, the agent's 'life support systems' cannot be frozen. Functions that is critical to the agent's roaming capability, such as basic communication module, unfreeze operation module (which requires an unfreeze key to execute), must remain functional when the agent is roaming. All other functions and data not critical to agent roaming can be frozen and subsequently activated when the agent reaches its destination.

### 2.2.5 Agent Transport

Once frozen, the agent is ready for transmission over the Internet. To activate roaming, the agent sends a request containing the roaming permit to the source receptionist. The source receptionist can optionally verify the validity and authenticity of the roaming permit. Since the roaming permit (as well as the entry permit inside it) will be inspected one more time when it reaches the destination receptionist, the inspection by the source receptionist is optional.

If the agent's roaming permit is valid, the source receptionist will transmit the frozen agent to the destination receptionist as specified in the entry permit. Once the transmission is completed, the source receptionist will terminate the execution of the original agent and make itself available to other incoming agents. The involvement of the source receptionist in the transport ends here.

### 2.2.6 Agent Pre-Activation

When the frozen agent reaches the destination receptionist, it will inspect the agent's roaming permit and the entry permit (contained in the roaming permit) carefully. By doing so, the destination receptionist can establish the following:

(1) The agent has been granted permission to enter the destination.
(2) The entry permit carried by the agent has not expired.
(3) The agent has obtained sufficient authorization from its owner/butler for roaming.
(4) The roaming permit carried by the agent has not expired.

If the destination receptionist is satisfied with the agent's credentials, it will activate the agent partially and allow it to continue agent transport process.

### 2.2.7 Request for Unfreeze Key and Agent Activation

Although the agent has been activated, it is still unable to perform any operation since all sensitive codes/data are frozen. To unfreeze the agent, it has to request for the unfreeze key from its owner/butler. To prove the authenticity of the destination, the destination receptionist is required to sign the random challenge in the roaming permit. The request for unfreeze key contains the session number, the certificate of destination and the signature on the random challenge.

The agent owner/butler can verify that the agent has indeed reached the right destination by validating the signature. If the signature is valid, the agent owner/butler will retrieve the unfreeze key based on session number, encrypt it using the destination's public key and returns it to the agent.

The destination receptionist can decrypt the unfreeze key using its private key and passes the unfreeze key to the agent. Using the unfreeze key, the agent unfreezes itself. To prove to the destination host that the incoming agent is indeed the agent requesting the entry permit, the agent will use its private key to sign the random challenge in the entry permit and return it to the destination receptionist. Once this signature has been verified, the destination receptionist fully activates the agent so that it can continue its execution in the new host.

The direct agent transport process is completed.

## 2.3 Unsupervised Agent Transport

Supervised agent protocol is not a perfect solution to agent transport. Although it provides tight supervision to an agent owner/butler, it has its limitations. Since the agent owner/butler is actively involved in the transport, the protocol inevitably incurs additional overhead and network traffic. This results in lower efficiency of the protocol. This is especially significant when the agent owner/butler is located behind a network with lower bandwidth, or the agent owner/butler is supervising a large number of agents. In order to provide flexibility between security and efficiency, unsupervised agent transport is proposed. The steps involved in unsupervised agent transport are shown in Figure 2.

### 2.3.1 Request for Entry Permit

In supervised agent transport, session ID and key pair are generated by the agent butler. However, for unsupervised agent transport, these are generated by the destination receptionists because agent butler is no longer on-line to the agents.

### 2.3.2 Pre-roaming Notification

Unlike supervised agent transport, the agent does not need to seek for explicit approval to roam from its owner/butler. Instead, a pre-roaming notification is sent to the agent owner/butler through indirect means. It serves to inform the agent owner/butler that the agent has started its roaming. The agent does not need to wait for the owner/butler's reply before roaming.

### 2.3.3 Agent Freeze

Agent freeze is very close to the same step under supervised agent transport, only that the encryption key is generated by destination instead of agent butler.

### 2.3.4 Agent Transport

This step is the same as that in supervised agent transport protocol.

#### 2.3.5 Request for Unfreeze Key

The identification and verification processes are the same as compared to supervised agent transport, the exception being that the unfreeze key comes from destination receptionist.

#### 2.3.6 Agent Activation

This step is the same as that in supervised agent transport.

#### 2.3.7 Post-roaming Notification

Upon full activation, the agent must send a post-roaming notification to its owner/butler. This will inform the agent owner/butler that the agent roaming has been completed successfully. Again, this notification will take place through an indirect channel so that the agent does not need to wait for any reply before continuing with its normal execution.

### 2.4 Bootstrap Agent Transport

Both supervised and unsupervised agent transport make use of a fixed protocol for agent transport. The procedures for agent transport in these two protocols have been clearly defined without much room for variations. It is realized that there exist applications that require special transport mechanism for their agents. For example, applications that involve highly sensitive content may wish to use a proprietary protocol for their agent transport. In order to allow this flexibility, SAFER provides a third transport protocol, bootstrap agent transport. Under bootstrap agent transport, agent transport is completed in two phases. The first phase is to send a transport agent to the destination using either supervised or unsupervised agent transport. In the second phase, the transport agent takes over the role of destination receptionist and continues the transport of its parent agent with its own agent transport protocols. In this way, different applications can implement their transport agents using the preferred transport mechanisms and still be able to make use of the SAFER agent transport. Bootstrap agent transport is illustrated in Figure 3.

In the first phase, the transport agent is sent to the destination receptionist using either supervised or unsupervised agent transport with some modifications. The original supervised and unsupervised agent transport requires agent authentication and destination authentication to make sure that the right agent reaches the right destination. Under bootstrap agent transport, the transmission of transport agent does not require both agent authentication and destination authentication.

Once the transport agent reaches the destination, it starts execution in a restricted environment. It is not given the full privilege as a normal agent because it has yet to authenticate itself to the destination. Under the restricted environment, the transport agent is not allowed to interact with local host services. It is only allowed to communicate with its parent until the parent reaches destination. A maximum time frame is imposed on the transport agent during which the transmission of its parent must complete. This is to prevent the transport agent from hacking attempts to local host. SAFER allows individual transport agents be customized to use any secure protocol for parent agent transmission. Concerns such as anonymity, secrecy, integrity etc should be taken care of by the transport agent. If the algorithm used by the transport agent is not secure, the whole agent may be compromised. In SAFER, parent agent assumes the responsibility of making sure its transport agent uses a secure transport protocol.

When the parent agent reaches the destination, it can continue the handshake with the destination receptionist and perform mutual authentication directly. The authentication scheme is similar to that in supervised/unsupervised agent transport.

## 3. Implementation

To prototype the design of agent transport, the three protocols discussed above have been implemented.

The prototype is built on Windows 95/NT platform using Java (see screenshot shown in Figure 4). Since Java is a platform-independent language, the prototype can be deployed

to any other platform that supports JVM (Java Virtual Machine). There are a few reasons why Java is chosen as the implementation language. Firstly, the most powerful feature of Java – platform independence, makes it the ideal language for building Internet-based applications. In order to provide interoperability across multi-vendor platforms, a truly platform independent language is desired. With Java, the prototype can be build once and run anywhere on other platforms.

Furthermore, the garbage collector feature of Java significantly reduces the programming effort and allows developers to concentrate on programming logic rather than taking care of memory. Unlike some other languages such as C/C++, Java VM manages memory automatically through its garbage collector.

Another feature that benefits the prototyping is thread-safe. Java language makes it easy to develop multi-thread applications. Threading is taken care of by JVM so that applications using Java threading is automatically thread-safe. In other languages, extra effort is needed to ensure the program runs normally under multi-thread scenario.

As a first step in the prototyping, unsupervised agent transport has been implemented. Two agent receptionists are setup in different hosts simulating the source host and destination host. An agent carrying certain functions is invoked from the source host. It kicks off a series of message exchanges under unsupervised agent transport and eventually reaches the destination host. During the process, the source receptionist and destination receptionist are involved in the handshake. When the agent reaches the destination, it successfully unfreezes itself and is activated for normal execution. During the simulation, two indirect messages are sent to the agent owner/butler (pre-roaming and post-roaming notices) as stipulated in the unsupervised agent transport protocol.

Functions to be carried out by the agents are loaded into the agent body before roaming. They will be preserved throughout agent transport. All functions carried are classified into sensitive functions and non-sensitive functions. Examples of sensitive functions are digital signature generation, negotiation strategy, mission statement, etc. Sensitive

functions will be encrypted during the actual transmission. Non-sensitive functions refer to both functions with less sensitivity and functions that are vital to agent transport. Functions with less sensitivity do not need to be encrypted, and functions that are vital to agent transport cannot be encrypted otherwise it will not be able to perform regularly.

In the implementation, encryption on agent functions is done by first converting the agent function's byte-code into a binary stream (using the serialization feature of Java), and subsequently performing symmetric key encryption on the binary stream. The encrypted byte stream is carried in the agent body during agent transmission. When the agent reaches the destination, the encrypted byte stream will be decrypted into the original binary stream. From the original byte stream, the byte-code can be reconstructed and the agent function class can be dynamically loaded. The serialization feature of Java significantly reduces programming complexity here.

The flow of unsupervised agent transport protocol implementation is summarized below as an example:

1. Entry Permit Request (Agent to Source Receptionist)
   Message content: agent certificate, destination address, and purpose of visit description.
2. Entry Permit Request (Source Receptionist to Destination Receptionist)
   Message content: agent certificate, purpose of visit description.
3. Session Generation (Destination Receptionist)
   Action: Generate random session key, generate random challenge, generate freeze/unfreeze key pair, and store session information to local database.
4. Issue Entry Permit (Destination Receptionist to Source Receptionist)
   Message content: agent description and entry permit (content of entry permit is discussed in the earlier section).
5. Entry Permit Reply (Source Receptionist to Agent)
   Message content: entry permit.
6. Agent Freeze (Agent)

Action: generate random session key, encrypt sensitive functions with session key, encrypt session key with freeze key.

7. Pre-Roaming Notice (Agent to Agent Owner/Butler – Indirect)
   The notice is sent as an email message with destination address in the message body.
8. Send Request (Agent to Source Receptionist)
   Message content: entry permit, destination address and encrypted agent.
9. Send Agent (Source Receptionist to Destination Receptionist)
   Message content: entry permit and encrypted agent.
10. Partial Activation (Destination Receptionist to Agent)
    Action: activate the agent for execution to finish the agent transport process.
11. Unfreeze Key Request (Agent to Destination Receptionist)
    Message content: agent certificate, entry permit, and session identifier.
12. Load Session (Destination Receptionist)
    Action: validate entry permit, load unfreeze key from database based on session identifier.
13. Unfreeze Key Reply (Destination Receptionist to Agent)
    Message content: unfreeze key.
14. Unfreeze and Activation (Agent)
    Action: decrypt session key using unfreeze key, decrypt sensitive functions using the session key.
15. Post-Roaming Notice (Agent to Agent Owner/Butler – Indirect)
    An email is sent out to agent owner/butler notifying the success of agent transport.

## 4. Conclusions

SAFE is designed as a secure agent architecture for e-commerce. The foundation of SAFE is the agent transport protocol, which provides intelligent agents with roaming capability without compromising security. General security concerns as well as security concerns raised by agent transport have been carefully addressed. The design of the protocol also takes into consideration differing concerns for different applications. Instead of standardizing on one transport protocol, three different transport protocols are

designed, catering to various needs. Based on the level of control desired, one can choose between supervised agent transport and unsupervised agent transport. For applications that require high level of security during agent roaming, bootstrap agent transport is provided so that individual applications can customize their transport protocols. The prototype of SAFE agent transport protocol has been developed and tested.

Agent transport protocol provides the secure roaming capability to SAFE. With a secure agent transport protocol, agents in SAFE can roam from host to host without being compromised. However, this does not complete the security framework in SAFE. Agent transport protocol only addresses the security issues involved when the agent is roaming. There are other security issues to be considered. One of them is to protect agents against malicious hosts as well as protecting a host from malicious agents. To address these issues, agent flight recorder (AFR) and SAFE certification are being proposed. The design of AFR and SAFE certification will be studied in greater details in the near future in order to complete the security framework for SAFE.

As an evolving effort to deliver a more complete architecture for agents, SAFER (Secure Agent Fabrication, Evolution and Roaming) architecture is being proposed to extend the SAFE architecture. In SAFER, agents not only have roaming capability, but can make electronic payments and can evolve to perform better.

**References**


B. Schneier (1996). Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd Ed., John Wiley & Sons, Inc, New York.

C. Guilfoyle (1994), Intelligent Agents: The New Revolution in Software, OVUM, London.

C. Thirunavukkarasu, T. Finin, and J. Mayfield (1995), Secret Agents – A Security Architecture for the KQML Agent Communication Language, CIKM'95 Intelligent Information Agents Workshop, Baltimore.

D. E. White (1998), A Comparison of Mobile Agent Migration Mechanisms, Senior Honors Thesis, Dartmouth College.

D. Johansen, K. Marzullo, and K.J. Lauvset (1999), An Approach towards an Agent Computing Environment, ICDCS'99 Workshop on Middleware.

D. Kotz, R. Gray, S. Nog, D. Rus, S. Chawla, and G. Cybenko (1997), Agent Tcl: Targeting the Needs of Mobile Computers, IEEE Internet Computing, Vol. 1, No. 4, pp. 58 – 67.

D. Rus, R. Gray, and D. Kotz (1996), Autonomous and Adaptive Agents that Gather Information, AAAI '96 International Workshop on Intelligent Adaptive Agents.

D. Rus, R. Gray, and D. Kotz (1997), Transportable information agents. In Michael Huhns and Munindar Singh, editors, Readings in Agents. Morgan Kaufmann Publishers, San Francisco.

D. Westhoff (2000), On Securing a Mobile Agent's Binary Code, Proceedings of the ICSC Symposia on Intelligent Systems and Applications (ISA'2000).

E. Z. Bem (2000), Protecting Mobile Agents in a Hostile Environment, Proceedings of the ICSC Symposia on Intelligent Systems and Applications (ISA'2000).

F. B. Schneider (1997), Towards Fault-tolerant and Secure Agentry, Invited paper, 11th International Workshop on Distributed Algorithms, Saarbrücken, Germany.

J. B. Odubiyi, D. J. Kocur, S. M. Weinstein, N. Wakim, S. Srivastava, C. Gokey, and J. Graham (1997), SAIRE – A Scalable Agent-Based Information Retrieval Engine, Proceedings of the Autonomous Agents 97 Conference, Marina Del Rey, California, U.S.A., pp. 292 – 299.

R. Gray (1997), Agent TCL: A Flexible and Secure Mobile-agent System, Ph.D. thesis, Dept. of Computer Science, Dartmouth College.

R. Schoonderwoerd, O. Holland, and J. Bruten (1997), Ant-like Agents for Load Balancing in Telecommunications Networks, Proceedings of the 1997 1st International Conference on Autonomous Agents, Marina Del Rey, California, U.S.A., pp. 209 – 216.

S. Corley (1995), The Application of Intelligent and Mobile Agents to Network and Service Management, et al. 5th International Conferences on Intelligence in Services and Networks, IS&N'98, Antwerp, Belgium, May 1998 Proceedings.

S. U. Guan, Y. Yang (1999), SAFE: Secure-roaming Agent For E-commerce, CIE'99, Melbourne, Australia, pp. 33-37.

T. Finin, J. Weber (1993), Draft Specification of the KQML Agent Communication Language, http://www.cs.umbc.edu/kqml/kqmlspec/spec.html.

T. Finin (1994), et al. KQML – A Language Protocol for Knowledge and Information Exchange, CS Tech. Report CS-94-02, University of Maryland.

Thomas Sander, Christian F. Tschundin (1998), Protecting Mobile Agents Against Malicious Hosts (1998), Mobile Agents and Security LNCS 1419, pp. 44 – 60.

**Figure Captions**

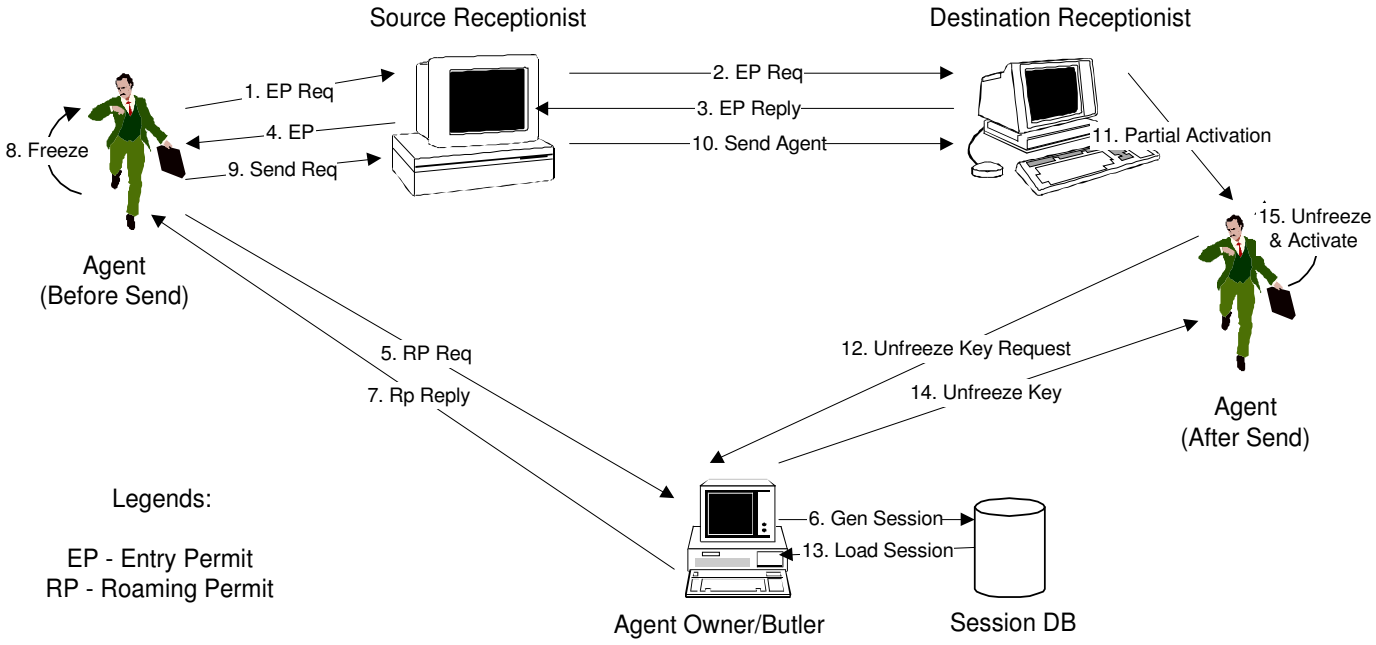


Figure 1: Supervised Agent Transport

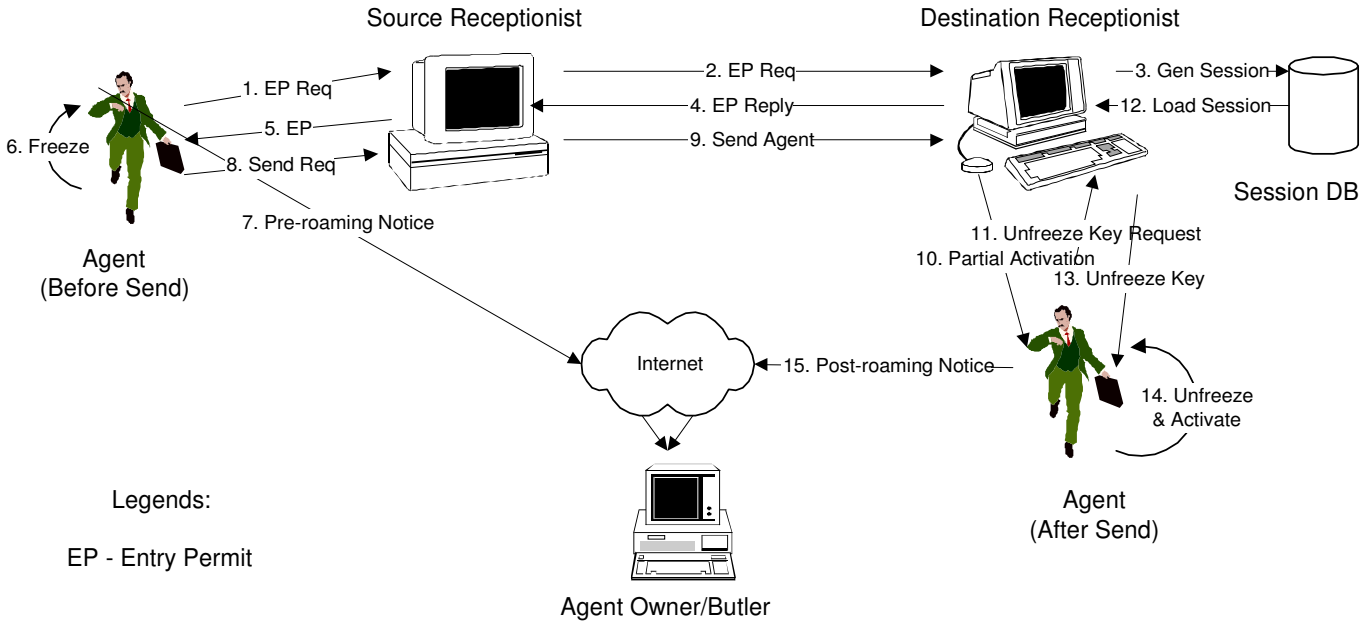


Figure 2: Unsupervised Agent Transport

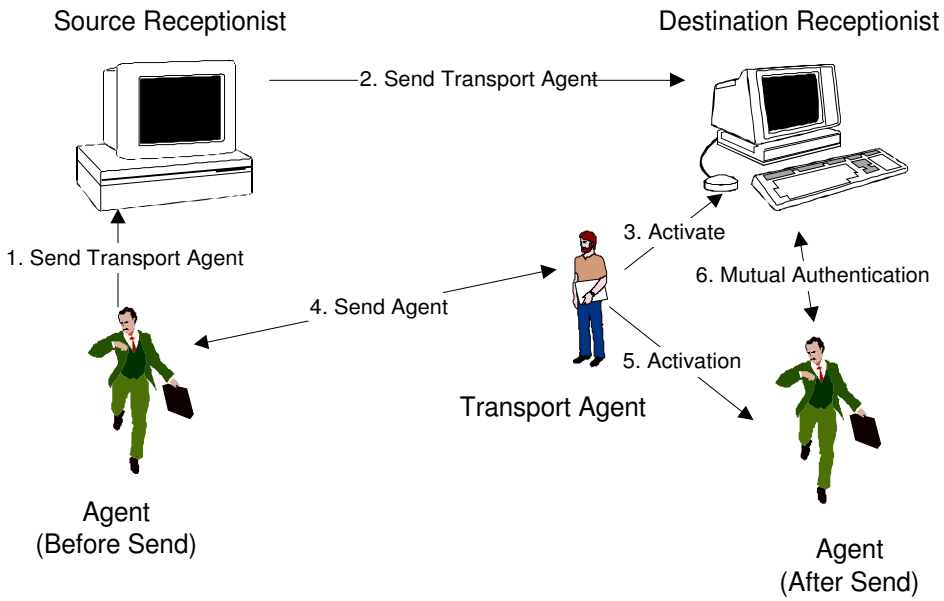


Figure 3: Bootstrap Agent Transport

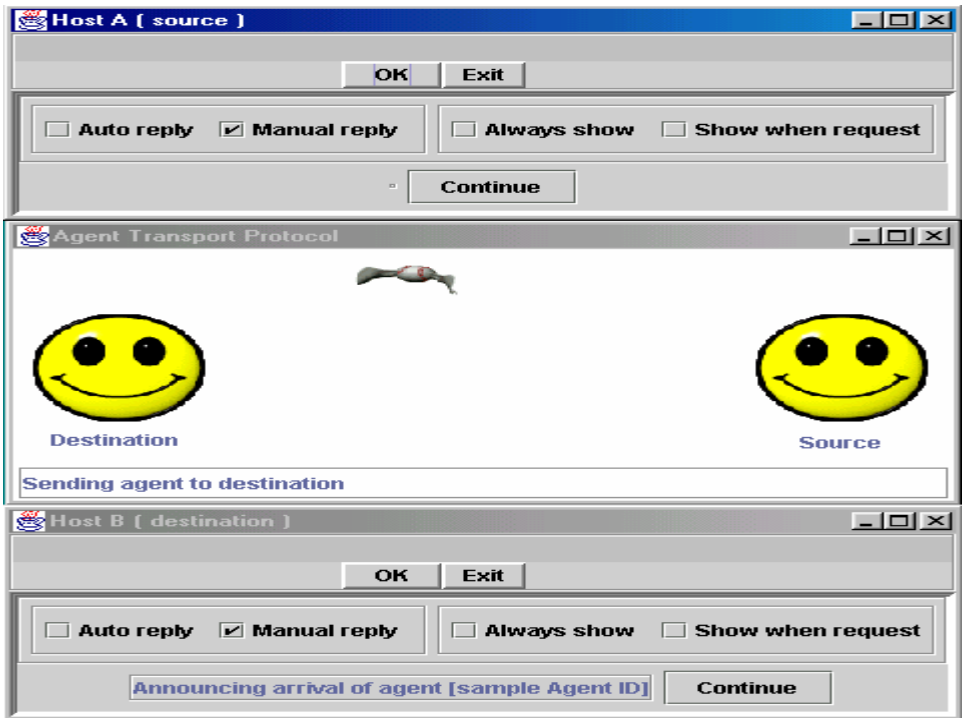


Figure 4. Screenshot of Agent Transport Protocol