Active Network Approach to the Design of Secure Online Auction Systems

Basem Shihada and Sampalli Srinivas

Faculty of Computer Science
Dalhousie University
Halifax, NS, B3H 1W5, Canada
{shihada , srini} @cs.dal.ca

ABSTRACT

Online auction systems require high-speed bid transmission, large bandwidth and maximum bid security. Most current implementations perform all auction operations on the auction server without any support from the network. This results in a large number of bid collisions, thus reducing the performance. Furthermore, many current implementations do not provide security measures such as client authentication and authorization, bid validation and bid data encryption.

This paper presents a novel approach to the design of secure online auction systems using active networks[1]. Active Networking is a new networking paradigm that inserts intelligence within the network by offering dynamic programming capability to network routers as well as to packets traveling in the network The goal of this design is to develop efficient and portable quality of service mechanisms to perform a variety of auction system tasks in a secure manner. The paper discusses the system design model and its implementation on the active network test bed. The utilization of active network technology accomplishes several enhancements such as the elimination of bid collisions, better server utilization, distribution of the security process, adjustment of the server bandwidth and enabling of monitoring the auction rules. Performance results from experiments on the test bed validate many of the advantages of the proposed approach.

KEYWORDS

Active networks, auction systems, bid security, quality of service (QoS), query certificate management (QCM), packet language for active networks (PLAN).

1. INTRODUCTION

The Internet and the World Wide Web have energized the already fast-moving world of commuting and created previously unthinkable opportunities for communication between computer users. The advent of Internet-based electronic commerce has resulted in a tremendous interest in the design and development of online auction systems. Such systems have played a vital role in expanding current business transactions to much higher levels by allowing a larger number of potential customers and companies to interact in a shorter time with lower costs. Furthermore, they allow businesses to access bidder information such as who buy, how they buy, when they buy, and what they buy.

Traditional online auction systems rely on the availability of advanced computing capability and high-speed network availability [1,2,3]. Although considerable work has been done to enhance the overall performance of online auction systems, they are still largely dependant on the organization of the average network bandwidth latency, the reduction of the management overhead, and the minimization of synchronization. Most current implementations perform all auction operations on the auction server without any support from the network. This results in a large number of bid collisions, thus reducing the performance. Furthermore, many current implementations do not provide security measures such as client authentication and authorization, bid validation and bid data encryption.

1.1 Active Networks

Active networking is a new networking paradigm that inserts intelligence within the network by offering dynamic processing and programming capability to network nodes and packets traveling in the network [3,4,5]. Routers in active networks are able to perform computations up to the application layer. The result is a more flexible and powerful network that can be used to speed up the deployment of new applications. Applications that can benefit from active networks include network management, congestion control, multicasting and caching.

Several architectures have been researched for deploying active networks [5]. Three kinds of active network architectures have emerged: *active packets*, in which the computations are limited to the packets traveling in the network; *active nodes*, in which the computational power comes from the nodes only; and the *hybrid version* which combines both active packets and active nodes approaches. According to [5], the hybrid architecture appears to be the most promising one. Among the different active network frameworks that have been proposed, PLAN (Packet Language for Active Networks) has emerged as an important hybrid architecture [6,7,8]. PLAN combines security, flexibility, efficiency and provides an execution engine for developing active network applications.

[1] This research was supported by a grant from the Canadian Institute for Telecommunications Research (CITR) under the NCE program of the government of Canada.

1.2 Proposed Work

This paper presents a novel approach to the design of secure online auction systems using active networks. Given *n* number of auction users on *k* different online servers each with *p* number of different items, we wish to develop efficient and highly portable secure mechanism to perform various client needs. Efficiency is a performance measure to indicate the effective utilization of bandwidth within a network. Portability refers to how easily the auction system can be implemented independently from any low-level primitives like network and operating system environments. Security refers to client authentication and authorization, bid validation and bid data encryption.

Some initial work has been done in the area of utilizing active networks to auction systems [3]. However, our approach integrates security and distributed databases into the design of online auction systems. The utilization of active network technology accomplishes several enhancements such as the elimination of bid collisions, better server utilization, distribution of the security process, adjustment of the server bandwidth and enabling of monitoring the auction rules. Performance results from experiments on the test bed validate many of the advantages of the proposed approach.

The rest of the paper is organized as follows. Section 2 describes the system design model of the active network architecture. Section 3 provides details of implementation of internal services and the security services. Section 4 describes the performance results of experiments run on the test bed and Section 5 provides concluding remarks, summarizing the features of the design approach.

2. SYSTEM DESIGN MODEL

This active auction system is designed based on the concept of client/server approach with stream sockets. The system consists of an auction server application that allows multiple clients to connect to the server and begin the auction session. As the server receives each client connection, an instance of the auctioneer is created to process the client in a separate thread of execution. The server maintains the client information and the auction session information, so it can determine if the client information and bids are valid. Each client application maintains its own version of auction interface on which the state of the auction session is displayed. The clients can only place one unique value in the bidding field.

When the active auction server starts, it begins accepting client connections. Then, a new client will be added to the connected client table. The server starts checking whether this client is an already registered or a new client. If it is a new client a new process will start to collect the client information. If it is an already registered client the auctioneer creates a process to negotiate the authentication and authorization process through the input and output streams. The client method controls the information that is sent to the active routers and the information that is received from the server. The server acknowledges the client connection, identifies the client, identifies the selected items, and updates information about the client and the auction session.

The Client/Server application implements multi-threading task operations so that a separate thread is used to continually read messages that are sent from the server to the client. The methods that are responsible for validating the client bids are all kept synchronized, to prevent more than one client from modifying the state information of the item price simultaneously.

Figures 1 & 2 illustrate the core design of the online auction system using active network approach. The model has been extended from the basic active network architecture presented in [5]. Basically, the purpose of the active node is to work as a computationally fast filter applied to all bid packets, only the bids that pass through the filter update the item price.

Our design uses a database distribution mechanism, using this mechanism its possible to reach an acceptable network usability and speed. The auction database server distributes the items over the active nodes. The auction system can be seen as a collection of active nodes. Each node holds the auction rules, auction services, and part of the database. This part is dynamically located over the nodes. The items can be added, removed or changed depending on the use of the clients. Furthermore, if each client wishes to bid on several items, there will be no need for overloading one router with the whole database or a big part of the database. Usually, this technique will enable multiple connections with the in-between network devices. In the auction applications this matter will not be a big concern, because the client will have his/her connections for a short time, which will be the bidding session.

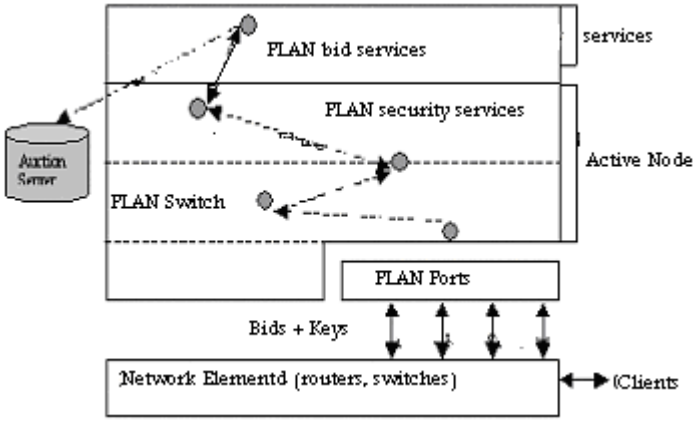

Fig.1: Active node middle-ware structure

Management functionality is added for both performance and security reasons. For example, if a client entered an auction session for a certain item, there would be no need at all for the server to provide this client with any information about the other ongoing auction sessions or other items. Similarly, if the active node checked the client information and found that this client was trying to break the auction rules, the active node will

directly warn, suspend or disconnect this client without loading the server or effecting the auction session.

The high level Client/Server application communicates with the active network using PLAN ports. PLAN ports listens for connections from both the active router and the host application [6,7]. All packets sent from the host application to the active router will be called active packets. Messages created from within the active node are service dependent, so its purpose is to do the required filtering over the active packets.

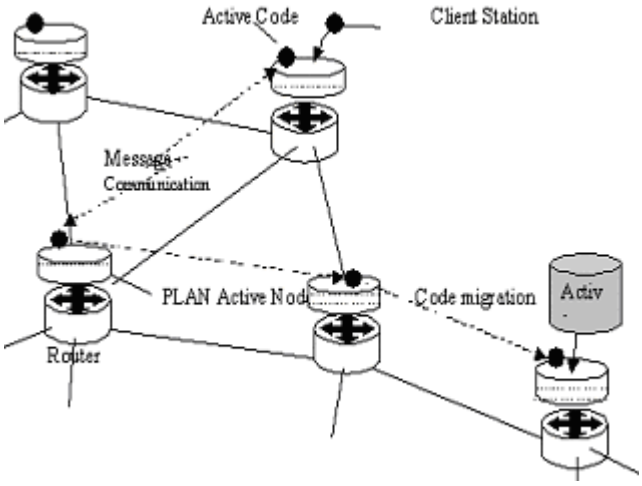

Fig.2: Architecture view of active online auction system

Security (authentication and authorization) is applied using two levels of hierarchy. First, the host application, which will inject PLAN programs to the network, must authenticate itself to the active node using public keys. Second, the client uses the shared secret key to either digitally sign or encrypt the bidding data. Furthermore, for the client host application to be connected to the server, it first must go through some active nodes. The client host application asks the active node for an authentication. Then the active node injects a PLAN program to call a remote service to exchange the keys and identify a selected key exchange protocol. When the active node receives the request, it first verifies the signature with the key to make sure that the certificate is still valid (the certificate has exchange keys, public key, client host application keys, and both addresses). The active node sends its public key to the client and stores the key sent from the client host application. The client receives a signature and does the same process that the active node has done.

The active node receives the approved message from the client host application and repeats this action with each incoming packets [7,8].

3. IMPLEMENTATION DETAILS

3.1 *Internal Services Implementation Model*

PLAN auction routers are designed to handle the auction tasks individually. All results will be returned back to the server and some will be shared with clients. Communication between the routers and clients is accomplished via through PLAN programs [8]. Figure 3 shows the network topology of the experimental test bed. Four independent PLAN routers are running on each host.

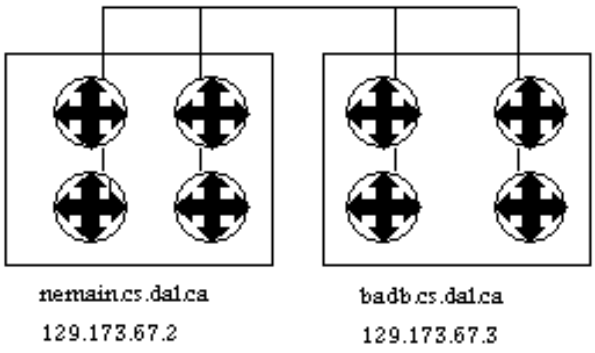

Fig.3 Network Topology over the test bed

In addition to the core services available to PLAN, additional services have been implemented to support the online auction system tasks. The following is an overview of how these services are designed and implemented [7,8,9].

Store Data to the PLAN active node:

The auction server invokes this service as soon as the database is ready to be distributed and stored over the active nodes. A PLAN program is injected to deliver the item list to the active node. When SetDataList service is first invoked the active node checks if the list is not empty, then it identifies its size and allocates the required memory. Finally, the item list is stored as a global list located on the active node. However, if an empty list is received the size will be raise an exception [9,10,11,12].

Initialize Winner List on PLAN active node:

This service is invoked from the setDataList service. A new winner list is created. This list will have the item pin code and the user who bid the highest for this item [9,10,11,12].

Get the Database from PLAN active node:

This service is only called from the auction server application. At the time the service is called, it will return back the data list from all active nodes. The server checks the latest prices of items at different auction sessions. This service is required to monitor the auction rules, check the item codes, validate the latest prices and make sure that the items have not been changed for any reason [9,10,11,12].

Get Winners from PLAN active node:

This service is only called from the auction server application. If this service is called it will return a PLAN list. This list will

contain the winner names and their items. The server either displays or saves this list for any future contacts with clients. This service is used to monitor the items, the winners, know what items are interesting to the clients and on which active node they are located [9,10,11,12].

Get Latest Item Prices from the active node:

This service is only called from the client application. Using any valid item code, the client can get the latest price of that item. If this service is called with a valid item code, it will return the item latest price. The item price will be displayed on the clients GUI. This service is used to monitor the items and keep the client updated with latest item price. This service has been designed to be a client on demand operation; this will reduce the number of operations on the active node [9,10,11,12].

Update Item Price on the active node (non-secure version):

This service is called from the Client application. Using a valid item code, the client can bid on any item he/she likes. The non-secure version of this service takes three arguments, the item code, the user name (the name is very important to keep the active node intelligent enough to know the last person made the bid), and the bid value. The service checks whether this item exists on the active node or not. If it exists and the received bid is higher than the current value of the item, the price is changed and true is returned to the user. Otherwise false is returned. This service is considered the bottleneck of the system. In other words, this service will be the called the most. The accepted bid will cause the updateWinner service to be called with two arguments, the item code and the client name to add an entry to that client indicating that he/she is the latest winner of this item.

3.2 Security Implementation Model*:*

When connected to the Internet, the system will be connecting to many unknown networks and their users. The three issues of authentication, authorization and confidentiality must be addressed.

3.2.1 Authentication:

Authentication is applied using two levels of hierarchy. First, the host application injecting PLAN programs must authenticate itself to the active node. Second, a shared secret key is created to either digitally sign or encrypt the transmitted data. To accomplish the above, Diffie-Hellman message exchange protocol has been implemented on our system using PLAN certificates [13,14,15]. The PLAN certificate has the following format:

Byte[] signer, Public key which has th following format (P:<digits>,Q<digits>,ALPHA<digits>,Y<digits>)
Byte[] cookie, random number to mix it.
Lone not_Valid_before, time to live
Long not_Valid_after, time to live
Int Exchanged_ID used to create the private key
Host PlanD_Address destination
Host Application_Address, source
DhmessageOne(certificate,digital signature), DhmessageTwo (certificate, digital signature), DhmessageThree(certificate, digital signature), and authEval(program, digital signature) are the security services that have been deployed.

3.2.2 Authorization:

The design considers every application that tries to access the active node as un-trusted application. In other words, the active node must keep a map of the principal users such as client applications and server applications. The implementation uses the PLAN QCM with few changes. The changes consist of adding the auction services that are used by the server (setDataList, getDataList, setWinner, getWinner, addItem, removeItem) and the services that are accessed by the client (updateData, updateWinner use, getNewPrice) [15,16,17].

The authorization process is activated as soon as a host application (Client/Server) tries to access the active node services. The active node receives the application public key and checks with QCM. If the key is in the sets of keys, the active node checks for eligibility to using the required services [17,18]. If the service is defined under this public key, then it will be accessed. Otherwise, a service not present exception will be thrown.

3.2.3 Encryption:

An online auction system should protect the bid unit (item PIN code, user login name and the bid value). In this design, an open-source encryption scheme (Cryptix toolkit[2]) is used to encrypted/decrypt the bid units between the host application and the active node [20,21]. Furthermore, the encryption scheme is implemented with features to prevent replay attacks (bi-directional data encryption is provided) [19,20]. At the client side, the Cryptix toolkit is capable of generating the triple-DES key, read the bid data and encrypt the client bid using the ECB mode. At the router side, the toolkit decrypts the received cipher text and writes the plaintext to the C service to continue the bid update process.

This cryptographic mechanism ensures that the bids will remain private and secure even if the auction system became more widespread. This design provides two different levels for bid protection. The auction system administrator can select data protection using either digital signature or data encryption.

4. PERFORMANCE RESULTS

Experimental results presented in this section have been done on the test bed set up in our lab for these experiments and research. It currently has five IA32 systems: three active PIII 450 128MB nodes running Debian Linux and two multimedia workstations running Windows NT4. Active network execution engines PLAN, ANTS and NetScript are available on the three active nodes. VPN tunnels can be created using IPSec Free S/WAN and PPTP (Point-to-Point Tunneling Protocol). AnetD provides connection to the ABone and Internet access is via the Dalhousie network. The performance of this design is a

2 Cryptix 3 is a cleanroom implementation of Sun's Java Cryptography Extensions (JCE) version 1.1.

reflection of the overall system transaction time, bid latency, bid collision, routers usage (CPU, Memory) and the data encryption time. Several random item PIN codes, item prices, user names, session periods are used to track the probability of system errors as will as bottleneck situations.

Figure 4 graphs the numerical values, which resulted from the total client transaction time. The total time is the sum of the bid encryption time, the network routing time, and the in-router data processing. It shows that the total biding time is almost similar and ranges between 490 to 530 milliseconds. On the other hand, the server in this case will not perform any transaction.

Figure 5 illustrates the client bid encryption time. Using different sizes of item codes and user names its shown that the encryption time ranges between 10 to 20 milliseconds. This test gives an indication of how fast the Cryptix 3.2 toolkit encryption engine over the system hardware. The graph peaks might have been caused because of the different sizes of the bid data or due to the internal functionality of the Cryptix toolkit.

Figure 6 illustrates how much CPU usage the router used through different auction conditions (e.g. at one stage the active router has received 12 different bids at the same time. The CPU usage has reached 4%).

Figure 7 illustrates how much memory usage the active router used during several auction session periods. It can be seen that the active router used an average of 3% of total memory (64 MB). Since the size of the sent and received data is small, encryption and decryption processing is fast. It takes on average between 19 to 23 milliseconds to complete the encryption operation. On the other hand, it takes on average between 21 to 25 milliseconds to perform the decryption operation [20].

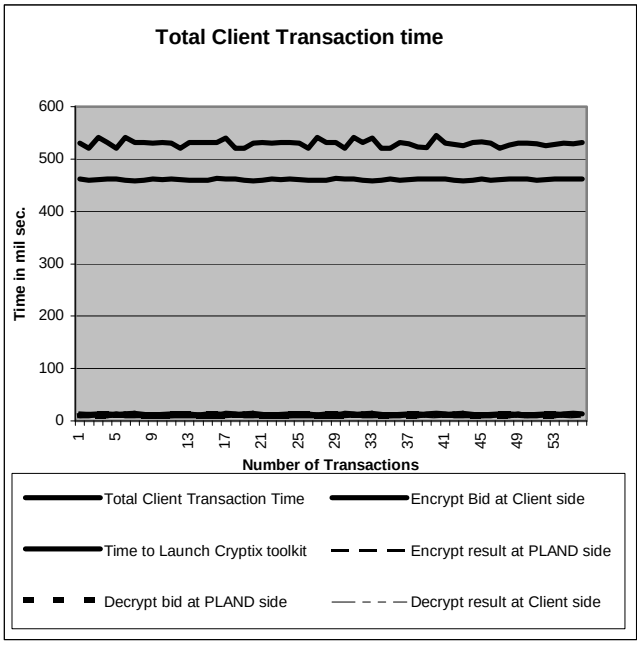

Fig. 4: Total transaction time

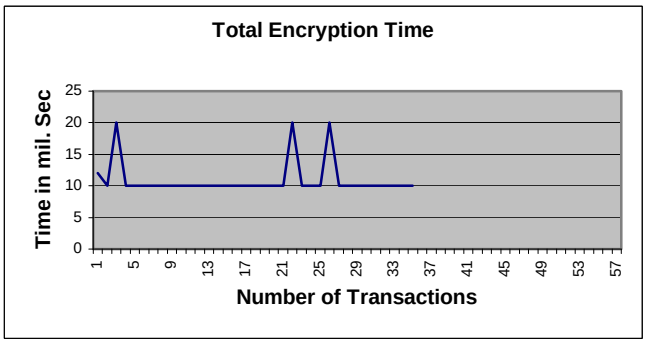

Fig. 5: Total Encryption Time

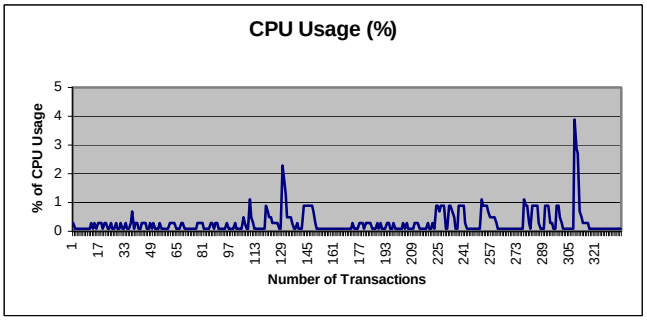

Fig. 6: % CPU Usage

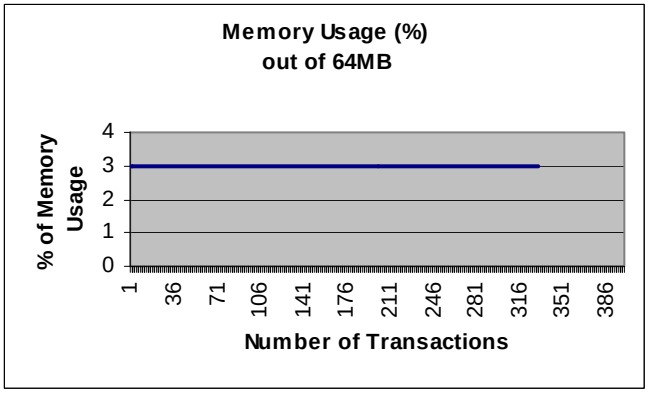

Fig. 7: % Memory Usage

6. CONCLUSION

The goal of this design is to develop efficient and portable quality of service mechanisms to perform a variety of auction system tasks in a secure manner. The utilization of active network technology accomplishes several enhancements such as reduction in number of bid collisions, better server utilization, distribution of the security process, adjustment of the server bandwidth and enabling of monitoring the auction rules

The implemented auction system has provided several advantages to the online auction systems; the following is a description of these features:

Elimination of bid collisions: The design provides a distributed pre-validation process that allows multiple bidders to send their

bids at the same time (bids not necessary to have the same value).

Expanding the reach of the server: using active routers, the control process of the server can cover a wider geographical area. The process of validating the trader bids is started and terminated inside the network using active routers. This expands the server reach allowing better server utilization.

Distributed Security processes: if a trader connects to the active node, the process of authentication and authorization will be activated. The router becomes a firewall to pre-identify the trader transactions. A QCM policy is added to control the privilege level of service access.

Server bandwidth adjustment: The active router checks and validates the bid reducing the server workload. Bids arriving below the actual price of the item and un-authorized traders are filtered within the active node.

Monitoring the Auction Rules: the active router takes care of who wants to withdraw from the auction session, checks the auction session time, checks the trader status and other auction rules. This allows the server to take care of the other tasks like monitoring the bidding agents (area of extension to electronic auction).

7. ACKNOWLEDGMENT

The authors would like to acknowledge the Canadian Institute for Telecommunications Research (CITR) for supporting this research. We would like to thank the major project leader Dr. Johnny Wong, University of Waterloo, and other collaborators and team members of the Resource Management in High Speed Networks project. Thanks to our team members.

REFERENCES

[1] Auction Types www.agorics.com/~agorics/auctions

[2] M.Kumar and S.Feldman, "Internet Auctions", IBM T.J.Watson Research Center White paper, 1999.

[3] U. Legedza, D. Wetherall, and J. Guttag, "Improving the Performance of Distributed Applications using Active Networks," MIT 1998.

[4] D.L. Tennenhouse et al, "A Survey of Active Network Research", *IEEE Communications Magazine*, January 1997, pp 80-86.

[5] K. Psounis, "Active Networks: Applications, Security, Safety, and Architectures", *IEEE Communications Survey*, 1999, Vol. 2, No.1.

[6] Official PLAN web site: can be found at http://www.cis.upenn.edu/~switchware/PLAN

[7] M. Hicks, P. Kakkar, J. T. Moore, C. A. Gunter, S. M. Nettles, "PLAN: A Packet Language for Active Networks", in *International Conference on Functional Programming (ICFP '98)*, Baltimore, MD.

[8] M. Hicks, P. Kakkar, J. Moore, C. Gunter and S. Nettles "Network Programming Using PLAN," *University of Pennsylvania. DARPA.*

[9] M. Hicks, "PlanD: The PLAN Active Router," v. 3.1. *DARPA.* 2000

[10] M. Hicks and P. Kakkar ,"The PLAN Tutorial," v. 3.2 . *DARPA.*

[11] M. Hicks, J. Moore, and P. Kakkar "PLANProgrammer's Guide," v. 3.2. *DARPA.* 2000

[12] M. Hicks, "PLAN Service Programmer's Guide," v. 3.2. *DARPA.* 2000

[13] C. Labonte, and S. Srinivas, "New Mechanisms for Extending PLAN Functionality in Active Networks,", *International working Conf. On Active Networks* (IWAN2000), Tokyo, October 2000.

[14] M. Hicks and A. Keromytis "A Secure PLAN," University of Pennsylvania. *DARPA.*

[15] M. Hicks, "PLAN Security Guide," v.3.2 *DARPA.*2000

[16] T. Jim, "QCM Reference Manual," *University of Pennsylvania*, 1998.

[17] A. Gunter, and T. jim, "Policy-Directed Certificate Retrieval (DRAFT)," *Univ. of Pennsylvania*, 1998

[18] D. Alexander, W. Arbaugh, and A. Keromytis. "Safety and Security of Programmable Networks Infrastructures," *University of Pennsylvania* 1998

[19] D. Alexander, W. Arbaugh, A. Keromytis and J. smith. "Security in Active Networks," *Univ. of Pennsylvania, 1999.*

[20] Cryptix 3.2 data encryption library works over java http://www.cryptix.org/products/cryptix31/index.html