Design and Implementation of a Secure Multi-Agent Marketplace

Ashutosh Jaiswal, Yongdae Kim, Maria Gini

University of Minnesota, Department of Computer Science and Engineering, 200 Union St SE, Minneapolis, 55455, MN, USA

Abstract

A multi-agent marketplace, MAGNET (Multi AGent Negotiation Testbed), is a promising solution for conducting online combinatorial auctions. The trust model of MAGNET is somewhat different from other on-line auction systems, since the marketplace, which mediates all communications between agents, acts as a partially-trusted third party. In this paper, we identify the security vulnerabilities of MAGNET and present a solution that overcomes these weaknesses. Our solution makes use of three different existing technologies with standard cryptographic techniques: a publish/subscribe system to provide simple and general messaging, time-release cryptography to provide guaranteed nondisclosure of the bids, and anonymous communication to hide the identity of the bidders until the end of the auction. By doing so, we successfully minimize the trust on the market as well as increase the security of the whole system. The protocol that we have developed can be adapted for use by other agent-based auction systems, which use a third party to mediate transactions.

Key words: Electronic Auctions, Multi-agent Systems, Security

1 Introduction

The business-to-business (B2B) e-commerce market is expanding and is expected to continue to grow. An online marketplace, used as a meeting point for B2B businesses, offers benefits to both buyers and sellers. For buyers, a marketplace can significantly ease the process of searching for and comparing providers, while for sellers a marketplace provides access to much broader

Email addresses: ashutosh@cs.umn.edu (Ashutosh Jaiswal), kyd@cs.umn.edu (Yongdae Kim), gini@cs.umn.edu (Maria Gini).

customer bases [16]. Sellers and buyers can simplify their operations by making use of auction-based marketplaces. Such marketplaces can be automated using software agents to represent the parties involved in the auction. Agents can either make autonomous decisions or assist their owners in their decision making process.

The MAGNET architecture provides support for complex multi-agent interactions [5]. MAGNET agents participate in auctions that are reverse auctions, since the auctioneer pays instead of getting paid; first priced, since the bids selected are the lowest cost feasible combinations of bids; sealed bid, since the auctioneer is the only one who sees the bids; and combinatorial, since bids can include multiple tasks with a single price for the combination. A multi-agent marketplace as MAGNET can thus effectively be used for carrying out contracting activities required for B2B transactions. However, in the absence of a secure architecture, its utilization in the real world has remained elusive.

Like most research projects, when the original MAGNET system was designed, security was not a major concern. However, as the system has evolved, it has become clear to us that in order for MAGNET to be used in real-world networks, a security architecture needs to be in place. Specifically, the security weaknesses of the original MAGNET design are in lack of secrecy of bids, non-repudiation, early bid opening, and bid manipulation. Such problems are quite common in auction systems. What makes MAGNET different from other auction systems is the presence of a trusted third party, the market, which acts both as a boon and a bane for security.

The presence of the market poses a unique challenge, that of ensuring the sanctity of the trust endowed in it. We came to realize that by ensuring this trust in the market we could overcome most of the existing security problems in MAGNET. Our solution is achieved by some modifications to the market architecture itself. The major modification is the use of a publish/subscribe system by the market to notify the agents about its actions. By cross-checking the actions taken by the market, other agents can ensure that the market is acting properly. Thus our notion of trust is dependent on the vigilance of other agents. In a similar manner, we can ensure that other agents are acting in a proper manner.

This paper is organized as follows. In Section 2 we examine the existing design of MAGNET and the resulting vulnerabilities. In Section 3 we present the security assumptions for MAGNET. We outline the proposed protocol in Section 4. In Section 5 we analyze the efficiency and security of the protocol that we presented. Implementation of the proposed protocol is discussed in Section 6. We compare our work with existing methods in Section 7. Finally, in Section 8 we present conclusions and talk about future work.

2 MAGNET and its Vulnerabilities

The motivation for coming up with a security model for MAGNET is to make it usable on public networks without compromising the data exchanged on it. MAGNET provides support for a variety of types of transactions, from simple buying and selling of goods and services to complex multi-agent negotiation of contracts with temporal and precedence constraints [5]. However, if such a system is to be used for carrying out transactions in a commercial enterprise world, it is imperative to have a reasonable security mechanism. This section reviews the current architecture of MAGNET and the potential vulnerabilities it poses.

2.1 Current Architecture

The current MAGNET architecture is composed of multiple agents and the market. The agents are self-interested agents, which attempt to gain the greatest possible profits from their endeavors. An agent can take one of two different roles: customer or supplier. A *customer agent* has to fulfill a set of tasks, and does so by soliciting help from *supplier agents*. An agent can act at the same time as customer in a negotiation and as supplier in another. The market is the meeting point for the agents, and the place which mediates all communications between agents. To accomplish its goal, the customer agent generates a plan, which is a collection of tasks with time and precedence constraints, and submits through the market one or more Requests For Quotes (RFQs) to suppliers. Any supplier agent who wants to bid will respond with bids. The customer agent after receiving the bids decides which bids, if any, to accept. Finally the winning supplier agents execute the tasks included in their winning bids [6]. Following is a more detailed description of these steps.

2.1.1 Planning

In this phase the customer agent selects a market which specializes in the types of product or service categories necessary to accomplish its goals. The agent then comes up with a plan which would fulfill its goal. While coming up with the plan, the customer agent takes into account the value of its goal, which in general is time-dependent, the component tasks needed, and their precedence and time constraints. Based on the plan, the agent generates one or more RFQs and forwards them to the market.

2.1.2 Bidding

In this phase the market sends out notifications to the associated supplier agents about the availability of a new RFQ. Supplier agents then contact the market and obtain the RFQ. Any supplier agent interested in the RFQ, can formulate a bid in response to the RFQ. The supplier agent then forwards the bid to the market for validation and delivery to the customer agent. The market can hold the bids until the deadline of the auction is reached, or forward them right away to the customer agent. On receipt of bids, the customer agent evaluates them and selects one or more bids, which fulfill its plan. The customer agent then sends out a bid acceptance notice to the respective supplier agents through the market. The market keeps a record of the bid acceptance and notifies the winning supplier agents of their bid acceptance.

2.1.3 Execution

During the execution phase, each supplier agent works on the tasks which the customer agent had approved in the previous phase, while the customer agent monitors the execution of these tasks. The customer agent can also re-plan and issue a new RFQ if the plan execution does not proceed according to its expectations. Once the supplier agent is done executing the plan, it notifies the customer agent. The customer agent then makes a payment (or a payment commitment) to the supplier agent. This payment is recorded by the market as well. The protocol is illustrated in Figure 1.

Currently the planning and bidding phase of the MAGNET system have been implemented, however, the execution phase has not been implemented. Hence in this paper we will consider only the first two stages of the protocol.

2.2 Vulnerabilities

The original design of MAGNET had not considered security issues. This leads to the vulnerabilities described below.

2.2.1 Secrecy of the bids

In a sealed bid auction, it is necessary that the bids are opened only after the end of the auction. The timing of the disclosure of the bid information is important. The MAGNET system is primarily designed to carry out first price, private, sealed bid, reverse, combinatorial auctions. To avoid collusion, it is necessary that the bid data from a supplier agent is not available to other supplier agents. In addition, MAGNET can be utilized for carrying out

a public auction as well. In such a case it is imperative that the bid data is not made available, even to the customer agent before the closing time of auction. In the current system, the customer agent receives all the bids through the market. In addition none of the communication in MAGNET uses encryption of any kind at present. Thus both the customer and the market (as well as any eavesdropper) have access to the bid data before the close of the auction.

2.2.2 Authentication

As pointed out in the earlier section, communications between agents in MAGNET do not use any kind of encryption. Thus it is possible for an eavesdropper to intercept the communication between the parties involved in an auction and modify its content. There are no means for assuring the suppliers that the RFQs they received were actually sent by the customer agent they claim to come from. At the same time the customer agent has no means of determining if the bid was actually submitted by the supplier agent, which the bid states it originates from.

2.2.3 Non-repudiation

Closely related to the problem of authentication is the issue of non-repudiation. In MAGNET, if the winning supplier agent declines to go ahead with the contract, there is no means of proving that it was indeed that agent who submitted and won the bid. Similarly, agents can repudiate any other communication, if this suits their purposes.

2.2.4 Prior opening of bids

As discussed before, the bids submitted should not be opened before the end of the auction period. MAGNET does not require that bids are not opened early, but early opening creates opportunities for counter speculation [4]. Also, an insider in the market can open and inform its collaborators of the contents of any bid. In MAGNET, the customer agent and the market both have access to all the bids before the end of the auction. If the customer decides to collude with a supplier agent, it would be harming itself, as it might lose out on a potentially better bid (see 3). However, the market can always collaborate with a supplier and give it available information pertaining to other bids. The collaborating supplier agent can thus modify its own bid to gain a competitive advantage over other agents. Thus we need a mechanism to prevent early opening of bids.

Furthermore, the information about the bidder itself can be important information to other bidders. Therefore, it is also necessary to have some kind of

anonymization mechanism which would make it impossible to determine the originator of a message.

2.2.5 Manipulation of closing time

In an auction system it is possible that an insider might manipulate the closing time of an auction in order to exclude some bids from the auction. In MAGNET, the customer is free to ignore any of the bids received, but it cannot extend the closing time of the auction without issuing a new RFQ. The market can, however, prevent bids sent out by suppliers from getting to the customer agent. It can also convey the closing time differently by modifying the RFQs on their way to the suppliers.

2.2.6 Fairness

At times an important security requirement of an auction system is fairness. In order to maintain trust in the auction system, it is necessary that the bidders be assured that their bids were given fair treatment before deciding the winning bidder. Authentication and non-repudiation are important to ensure the supplier agents their bids are transmitted to the customer agent without any modification, and to ensure the customer agents their RFQs are delivered intact to the suppliers. Since the auctions in MAGNET are private, fairness in winner determination is not an issue. *Hence our solution will not address this problem.*

2.2.7 Fault tolerance

Some of the above mentioned problems can be caused just because of a failure of the auction service or bidding process. In case of MAGNET, either the failure of the customer agent or the market can be responsible for the failure of the auction process. This, strictly speaking, is not a security hole but a problem which might lead to other security problems. We plan on addressing this issue in the future.

3 Security Assumptions

The current architecture, despite its security vulnerabilities, is an example of a unique approach to multi-agent interactions. In earlier multi-agent systems, agents communicated and contracted with each other directly. In most cases these negotiations were complicated by an environment in which there is no

mutual trust between the agents. MAGNET makes use of a trusted third party, the market, which is utilized by the agents to carry out all the transactions. Thus agents can utilize their resources towards plan execution and bidding, instead of negotiating in a chaotic manner. The main trust assumptions for the market are:

- It is responsible for conveying the RFQs from the customer agent to the interested supplier agents. It is also responsible for communicating the bids from the supplier agent back to the customer agent.
- It acts as a record keeper by keeping records of all the transactions and movement of RFQs and bids that take place through it. In case of dispute, the market will act as an arbitrator using the saved records.
- It is responsible for aggregating statistical data from the auctions and making them available to interested parties at a later period of time. These data may include average duration of tasks, average costs, reputation of suppliers, etc. We assume that this statistical aggregation is performed correctly by the market. How to do this fairly and securely is one of our future concerns.

The customer agent is responsible for initiating negotiations in the manner described earlier. We assume that the customer agent will not collude with any of the supplier agents, since this will defeat the purpose of conducting an auction. In such case, the customer would be only wasting time and resources in trying to contract through MAGNET (besides paying any fee that might be charged by the market for its services). Agents wishing to do so (possibly because of a preferred business relationship) can always communicate directly with each other but outside the MAGNET system. We assume that the customer agent communicates with the supplier agents only through the market and vice-versa. This assumption is necessary to ensure that avenues for customer-supplier collusion are discouraged and reduced. Furthermore, the market will keep records of all transactions being conducted which can be used to ensure non-repudiation. This assumption becomes especially important in case MAGNET is utilized for conducting a public auction.

4 Proposed Architecture

4.1 Building Blocks

Before we explain details of the protocol, we briefly introduce the notation used in this paper:

PK_c	Public-key of a customer C
SK_c	Secret-key of a customer C
K_a	Symmetric-key a
m	Message
$E_{PK_c}(m)$	Public key encryption of m using PK_c
$D_{SK_c}(m)$	Public key decryption of m using SK_c
$S_{SK_c}(m)$	Signature of message m using SK_c
$TE(m)$	Time-release encryption of m
$hash(m)$	Hash of m
$Publish(m)$	Publishing of m

We now briefly introduce three key techniques used in our solution.

4.1.1 Publish/subscribe systems

One of the major components that we use in our protocol is a publish/subscribe system. In such a system, publishers can publish messages under certain topics. Subscribers can subscribe to topics of their interest and are notified of new postings under those topics, which they can then examine. Topics can be classified hierarchically and the message content defined in a way deemed suitable by the users. Such systems minimize message duplication. They have the added benefit of allowing anonymous postings by publishers and subscribers [22]. In our proposed architecture the market hosts such a publish/subscribe system, to which both the customer and supplier agents have access. All published messages are signed by the originator, which can be verified by the agents accessing them.

4.1.2 Time-lock puzzles

Another cryptographic technique which we utilize is timed-release cryptography, also known as time-lock puzzles [21]. These methods provide a way of encrypting a message such that no one can decrypt the message until a substantial amount of time has elapsed. Good time-lock puzzles prevent the use of parallel algorithms for decryption. Assume A wants to encrypt a message m with a time-lock puzzle for a period of T seconds. A picks at random two large primes p, q and computes $n = pq, \phi(n) = (p-1)(q-1)$. She then computes $t = TS$ where S is the number of squarings modulo n per second that can be performed by the solver. Then A picks a long random key k for some

secure symmetric encryption scheme and encrypts m using k. Let us call the resulting cipher-text C_m. She then computes $C_k = k + a^{2^t} \bmod n$ for some random $a, 1 < a < n$. Since A knows $\phi(n)$, she can do this efficiently. The time-lock puzzle will contain (n, a, t, C_k, C_m). In order to extract m anybody would need to compute a^{2^t} and the only way to do this without knowing $\phi(n)$ is to perform t *sequential* squarings. The time delay ensured by this solution is not absolute real time, but some time period which depends on the CPU power of the solver.

4.1.3 Communication anonymizer

In absence of an anonymization technology, it becomes easy for an outsider, as well as an insider, to associate bids to the bidder. This may not be an important requirement in certain auctions, but in MAGNET this is necessary to reduce and discourage market-supplier collusion and to make the supplier's bid unlinkable until the end of auction. A peer-to-peer (P2P) anonymizing network like Tarzan [10] can be used for this purpose. Tarzan achieves its anonymity with layered encryption and multi-hop routing. First, a host running an application that wants anonymity chooses a group of hosts to form a path through the network. Next, this source-routing host establishes a tunnel using these hosts, which includes the distribution of session keys. Finally, it routes data packets through this tunnel. The end point of this tunnel is a Network Address Translator (NAT). This NAT bridges the hosts in Tarzan and the hosts that are not aware of Tarzan. Similarly, the NAT receives the response packets from the outside hosts and reroutes them back through this tunnel. Selection of the NAT from amongst the peers is done at random. In our protocol all the communication between the suppliers and the market is anonymized using a P2P network created by the suppliers.

4.2 Securing MAGNET

Based on the requirements mentioned before we now propose the following architecture to enhance the security of the MAGNET system. We outline our protocol into the following phases: contracting, planning, bidding, auction close, and winner determination. Fig 2 illustrates the protocol in detail.

4.2.1 Planning

The customer sends a signed RFQ to the market for publishing.

$$Customer \xrightarrow{S_{SK_c}(RFQ)} Market$$

4.2.2 Bidding

The supplier receives the RFQ through the publish/subscribe system. If interested, it generates a bid-message comprising of three parts:

(1) General Information (GI): consists of the RFQ number and a sufficiently long random number, $(RFQ\#, r)$.
(2) Auction-session key: a symmetric session key, K_a.
(3) Bid data: consists of the price quoted by the supplier, the list of tasks, the time-line for plan completion, GI, and supplier's public-certificate.

It then signs and encrypts the message and sends it to the market:

$$Market \xleftarrow{[(RFQ\#,r),TE\{E_{PK_c}(K_a)\},E_{K_a}\{S_{SK_s}(Bid)\}]} Supplier$$

For all the bid-messages received by the market, it publishes *(RFQ#,(r,hash(M)))* on the publish-subscribe system, where M is the bid-message sent by a supplier to the market. The supplier can also check the publish/subscribe system and verify that its bid was actually received and displayed by the market. The customer can then retrieve the bids from the publish/subscribe system. However, it cannot access the bid data unless it decrypts the time-release cryptography. Since the exact timing of a time-lock puzzle is difficult to determine [21], the supplier agent would construct a puzzle that would take the customer longer than the auction deadline to solve.

4.2.3 Auction close

Once the auction closes the suppliers would release K_a to the market in an encrypted form, along with the customer's copy:

$$Market \xleftarrow{E_{PK_m}\{SK_s(K_a,r)\},E_{PK_c}\{SK_s(K_a,r)\}} Supplier$$

The market would then pass on the customer's portion of the encrypted key:

$$Customer \xleftarrow{E_{PK_c}\{SK_s(K_a,r)\}} Market$$

This individual encryption is necessary so that no one except for the market and the customer can decrypt the bids.

4.2.4 Winner determination

The customer agent uses various algorithms to determine the winner from the bids it has received [2, 3]. From the suppliers' certificates embedded in the bids, it can use statistical data to assist in the winner determination process. Once the winner has been determined, it would use the market's white-board to notify the suppliers about this.

$$Customer \xrightarrow{S_{SK_c}(RFQ\#, r_{winner})} Market$$

Once the market publishes this result on the publish/subscribe system any supplier can check to see if it is the winner. The customer agent can do a cross-verification by examining the publish/subscribe system, in order to deter any wrong doing on market's behalf. The market can then carry out statistical aggregation by decrypting the bids using their respective auction-session keys.

5 Analysis

5.1 Efficiency

The protocol that we have proposed follows the original message exchange mechanism as closely as possible to avoid a major redesign of the existing system. However, by utilizing a publish/subscribe system, the need for acknowledgment generation for each message has been eliminated. The suppliers and the customers can independently verify the data received by the other party, in an asynchronous manner, thus leading to better utilization of their resources. Using an anonymization layer can add some delay in message propagation, but the benefits are significantly higher.

Encryption and decryption of data is usually the most computationally intensive task in a security protocol. We have tried to minimize the encryption of messages whenever possible. Thus, instead of encrypting the entire third section with time-release cryptography mechanism, we introduced a second section, since encrypting and decrypting K_a is computationally cheaper because of the smaller size of the data. The most computationally intensive task in our protocol is time-release decryption. However, this step is not needed as long as the customer waits for the auction to close and the supplier releases the auction-session key K_a.

Our motivation behind using a time-lock puzzle is mostly as a deterrent to prior opening of the bids and is not meant to impose a computational penalty

on the customer. However, in case the supplier fails to provide K_a (e.g. the supplier comes under a Denial of Service attack, or it refuses to send the key on purpose), the customer can proceed with time-lock decryption. In such a case, if the customer faces a resource crunch, it can seek the market's help in decrypting it. The market is likely to have more computing power than the customer or the supplier agents, since it provides the auction infrastructure, and thus can offer its resources for decryption to the customer. However, even after decrypting the time-lock puzzle the market would not be able to get to K_a since it will be encrypted by customer's public key PK_c (for the same reason, anyone who intercepts the bid cannot get to the bid data). Thus the customer can safely shift the burden of time-lock decryption on the market, if required.

We have tried to come up with a practical value of S for different clock speeds of a present day processor (Intel Pentium 4), which is presented in Fig 3. The processors that we tested with range in clock speeds of 1.2 GHz to 2.4 GHz. From these values we have determined corresponding values of $t = TS$ for a single day (86400s) shown in Fig 4. The software setup for the systems used to carry out the tests are the same as described in section 6.4.

5.2 Security

Our security architecture overcomes the vulnerabilities in the existing MAGNET system, and, at the same time, it tries to enhance the basic trusted third party model. We overcome the problem of secrecy of bids and the identity of the bidder by employing cryptographic techniques of anonymization and public-key encryption and decryption. By employing a time-release cryptography mechanism, we ensure that the bids are not accessible by any agent until the close of the auction.

We try to ensure the trusted third party model in MAGNET by including additional safeguards. By requiring that the market publishes all the data received from customers and suppliers on the publish/subscribe system, we enable cross verification of the data by the opposite party. The copies of the messages published by the market can be used to ensure non-repudiation in situations where the sender refuses to acknowledge an action.

Since the RFQs are publicly available for verification, it is difficult for the market to manipulate the closing time of the bids without being noticed by the customer. By making the market post the data related to all the bids received, we also ensure that it does not purposefully reject any messages.

As discussed before, prior opening of the bids is a form of collusion existing in current auction systems. By not making the auction session-key (and the

resulting bid data which it encloses) immediately available to the market, we limit the market-supplier collusion. There could still be a collusion between the market and a supplier, but the market would be able to provide bid data only for the supplier agents who are already colluding with it. In absence of bid data for all the suppliers, a market-supplier collusion would not be useful. A collusion between all the suppliers and market would be similar to collusion between all the suppliers. This, as we discussed earlier, is beyond the scope of our current effort. The only visible information prior to decryption of the bid is the *RFQ#* and the random number *(r)* generated by the supplier. By enclosing the public certificate of the supplier with the remainder of the bid data, we ensure that the identities of the supplier agents are not known to the customer or the market before the end of the auction. However, once the auction is closed, the customer agent can use the information about the suppliers' identity to make its decision based on past statistical data.

The proposed solution leaves room for a Denial-of-Service (DOS) attack. An attacker can always send junk bids through the anonymous channel without providing its keys after auction close (section 4.2.3). The customer (or the market) would need to solve all the time-lock puzzles to obtain bids and to provide non-repudiation of the bidder, which might be a waste of resources if the resulting bids are junk. This problem arises due to our policy of protecting the bidder's identity until the auction close. In other words, our current mechanism does not have the ability to check if the bidder is authorized to bid in the auction. Even though MAGNET follows the open auction model, we need to have some kind of access control mechanism to provide non-repudiation. One way to address this issue is to use group signatures [11]. If the bid message is signed using a group signature, it is guaranteed that the bidder is a member of the group as long as the signature is verifiable. If the solution of the time lock puzzle results in a meaningless bid, the group manager can "open" the group signature to trace the actual signer. If the market plays the role of the group manager, this opens up the possibility of market-supplier collusion. The market can open the signature prior to closing, and the bidder information can be released to other suppliers. This can be overcome by designating an independent trusted third party as the group manager. However a more elegant solution would be the use of "time release group signature" where the signature can be verified immediately, while the "open" operation can be allowed only after solving the time lock puzzle. Unfortunately, at the time of writing of this paper, there is no such practical signature mechanism in existence.

6 Implementation

The current implementation is done using C functions for the customer, supplier, and the market to carry out the requisite encryption and decryption op-

erations. These functions currently read the data which need to be processed from a file, carry out the requisite operations, and then write the resulting data back to a different file. These files can then be sent over to other agents using any suitable protocol like HTTP, SMTP, SOAP etc. Since not all the MAGNET's components are in place at the moment, such an approach leaves space for a seamless future integration with the resulting complete system. Following is a description of the functions that are implemented and how they relate to the proposed protocol.

6.1 Customer

The customer needs to read a RFQ, sign it with its key, and then output the signed message for transportation to the market. Later when it receives the bids from the suppliers, it needs to decrypt these bids and decide a winner from amongst them. In the end it needs to notify the market and suppliers about the winners, if there is any.

All this functionality has been implemented in the form of functions which read the RFQ from a file (which will be outputted from the RFQ generation functions of MAGNET), sign it with the customer's private key and then write out the signed RFQ back to a file. Similarly these functions provide the ability to read the encrypted bid from a file, decrypt it using the session key read in from another file (which is obtained only after an auction closes) and write out the decrypted bid to another file. Once the winner is determined, the RFQ number and the random number obtained from the winning supplier is concatenated and written to a file in order to be transported to the market.

6.2 Market

The market's responsibility is to publish and record all the messages that it obtains from the customers and the suppliers, so that other entities in the system can verify them. It also needs to be able to carry out time lock decryption on behalf of the customer in case it becomes necessary to do so.

The market can use these functions to verify the signature on the RFQ obtained from the customer. Once the RFQ is verified, it writes out the original signed RFQ to a file, which can be read by suppliers. The functionality of a publish/subscribe system has not been implemented yet owing to the absence of an actual market. It should be easy to create a web-service to carry out this functionality in future. The market can also verify a supplier's signature on a bid, create a hashed version of the bid, concatenated with the RFQ and supplier information and write it to a file to be used for future verification. In

presence of a publish/subscribe system, this information could be published on it.

After the close of an auction the supplier would release the session key used to encrypt the bids. The market can decrypt the message encrypted using its public key to obtain the session key, and also pass the same message to the customer, who can decrypt it using its own private key. Once the session key is obtained the market can decrypt the bid and store the information for future use to a file. On obtaining the winner notification from the customer, the market can verify customer's signature, store the information for future use, and pass the information to the supplier.

6.3 Supplier

The supplier needs to be able to verify and read a RFQ. After it has obtained the information from the RFQ it can generate a bid in response to the RFQ. This bid will be encrypted using a session key generated by the supplier, which in turn needs to be time-lock encrypted and then public key encrypted for the customer. It also needs to be able to verify the winner information to determine if it is the winner.

The supplier functions for RFQ and winner verification are similar to those for the market. However the bid and session key time-lock encryption function is unique to the supplier. Having generated the bid, the supplier writes out the bid to a file. This file can be transported to the market using any of the protocols mentioned above. The communication anonymizer functionality for suppliers has not been implemented yet.

6.4 System details

Implementation of the protocol was carried out in C using the libcrypto library, which is a part of the OpenSSL [26] package. For message transportation PKCS7 [12] and S/MIME [19] format were used. Advanced Encryption System (AES) [14] was utilized for symmetric encryption/decryption with a minimum key-length of 128 bits and a maximum key-length of 192 bits. The RSA [20] algorithm was used for public/private key encryption/decryption. The same algorithm was used to create public certificates for the entities in the system. SHA1 [13] was used as the message digest algorithm. The primary test machine was a 1.8GHz Intel Pentium-4m system, with 512 MB RAM running Gentoo Linux (kernel 2.6.3-r1). The version of the openssl package used was 0.9.7.

7 Related Work

Most of the related work has either been in the area of securing auctions or in agent security, without much overlap between the two. In the field of auction security majority of the work has been done on sealed bid auctions in which the auction outcome is made public. Most of these protocols require m auctioneers out of which at least n should be trustworthy (*threshold cryptography*). Franklin and Reiter proposed one of these earlier systems in which using a variation of the secret sharing scheme algorithm they try to reduce the trust on the auctioneer [9]. In their system the value of n is a third of m. The resulting system requires exchange of numerous messages making it highly inefficient. This system cannot be applied to MAGNET as there exists only a single instance of the market (auctioneer).

SAM [15] is a system in which the trust is shifted from the auctioneer to a hardware implemented secure co-processor [28]. The basic idea is to replace the auctioneer by a combination of hardware and software which can be trusted by all the parties involved in the auction mechanism. Members examine the source code and the executable of the auctioneer software, before it is embedded into the secure co-processor. Any tampering of the hardware results in its self-destruction. The system is similar to MAGNET in the sense that only a single auctioneer exists in the system. However, the market in MAGNET is not absolutely trusted as SAM is supposed to be.

Protocols in which the auctioneer is completely eliminated have also been proposed [1]. The idea behind such protocols is to let the bidders decide the winner themselves by splitting and distributing all the bids amongst all the bidders. Using a secret sharing scheme, the bids can be assembled only if all the bidders are willing to do so. As long as a single bidder does not collude with the other bidders, individual bids cannot be determined. In case all the bidders collude, the auction mechanism becomes an "open cry" auction. Once the bids are reassembled the winner can be determined by finding the highest bid. This system cannot be applied to MAGNET as the winner determination process is more complex than simply determining the highest bid [2].

Some work has also been done on developing secure auction protocols for combinatorial auctions [25]. The focus of the research was on finding an optimal solution to combinatorial auctions, using secure auction servers. A system which utilizes a temporarily secret bid commitment has also been proposed [24].

When it comes to securing agents research has been done in the field of secure agent matchmaking [8], whereas an agent is used to establish communication between different individuals with similar interests. All individuals run the same copy (or clone) of the agent code. This in essence means communicating

amongst various instances of the same agent having identical characteristics.

Research has also been done on constructing secure agent societies from the ground up [18]. The FIPA [7] agents standard body has proposed some standards in this direction [23] as well. At the same time there has been work conducted on adding security and trust to existing agent systems [27]. There has also been work, on using a secure agent to secure transactions between other agents [17].

However, none of the systems described above meet all the security requirements faced by multi-agent marketplaces like MAGNET in the manner we have described.

8 Conclusion and Future Work

In this paper we have presented ideas from recent work in security protocols for conducting secure electronic auctions using multiple agents. We have proposed a security architecture that would ensure the security of MAGNET and similar multi-agent marketplaces when used over public networks. Using various existing technologies, our protocol builds upon the trust model of the original marketplace and develops a system which has better methods of controlling fraud and deception.

We have implemented the proposed solution for use within the existing MAGNET system. Eventually we would like to release a secure version of the system to be used over the Internet. Once the market agent has been implemented we would also like to implement the publish/subscribe system and setup a communication anonymizer amongst the agents. We would also like to focus our work on establishing protocols for agent registration and payment collection in the execution phase once an auction has been closed. Including fault tolerance into the system, so that the auction does not fail because of the failure of the entities involved, is also a part of our future plans.

References

[1] Felix Brandt. A verifiable, bidder-resolved auction protocol. In *Proceedings of the 5th International Workshop on Deception, Fraud and Trust in Agent Societies (Special Track on Privacy and Protection with Multi-Agent Systems)*, pages 18–25, 2002.

[2] J. Collins and M. Gini. Performance of winner determination search in combinatorial auctions with time constraints. *Submitted to the First*

International Joint Conference on Autonomous Agents and Multi-Agent Systems, July 2002.

[3] John Collins, Güleser Demir, and Maria Gini. Bidtree ordering in IDA* combinatorial auction winner-determination with side constraints. In J. Padget, Onn Shehory, David Parkes, Norman Sadeh, and William Walsh, editors, *Agent Mediated Electronic Commerce IV*, volume LNAI2531, pages 17–33. Springer-Verlag, 2002.

[4] John Collins, Scott Jamison, Maria Gini, and Bamshad Mobasher. Temporal strategies in a multi-agent contracting protocol. In *AAAI-97 Workshop on AI in Electronic Commerce*, July 1997.

[5] John Collins, Wolfgang Ketter, and Maria Gini. A multi-agent negotiation testbed for contracting tasks with temporal and precedence constraints. 7(1):35–57, 2002.

[6] John Collins, Ben Youngdahl, Scott Jamison, Bamshad Mobasher, and Maria Gini. A market architecture for multi-agent contracting. pages 285–292, May 1998.

[7] FIPA. The foundation for intelligent physical agents. http://www.fipa.org/.

[8] L. N. Foner. A security architecture for multi-agent matchmaking. In *Second International Conference on Multi-Agent Systems*, 1996.

[9] M. Franklin and M. Reiter. The Design and Implementation of a Secure Auction Service. In *Proc. IEEE Symp. on Security and Privacy*, pages 2–14, Oakland, Ca, 1995. IEEE Computer Society Press.

[10] Michael J. Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, D.C., November 2002.

[11] M. Joye G. Ateniese, J. Camenisch and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270, Santa Barbara, California, USA, August 2000. Springer.

[12] RSA Laboratories. Pkcs #7: Cryptographic message syntax standard. Technical Report 1.5, RSA Labs, November 1993.

[13] National Institute of Standards and Technology (NIST). Secure hash standard. *FIPS Publication 180-1*, April 1995.

[14] National Institute of Standards and Technology (NIST). Advanced encryption standard (aes). *FIPS Publication 197*, November 2001.

[15] Adrian Perrig, Sean Smith, Dawn Song, and J. Doug Tygar. SAM: A flexible and secure auction architecture using trusted hardware. In *First International Workshop on Internet Computing and E-Commerce (ICEC'01)*, pages 170–170, April 2001.

[16] Charles Phillips and Mary Meeker. The B2B internet report – Collaborative commerce. Morgan Stanley Dean Witter, April 2000.

[17] He Qi, Katia Sycara, and T Finin. Personal security agent: Kqml-based pki. In *Proceedings of the 2nd International Conference on Autonomous Agents and Multi Agent Systems*, Minneapolis, MN,, May 1998.

[18] He Qi, Katia Sycara, and Zhongmin Su. Security infrastructure for software agent societies. In Christiano Castelfranchi and Yao-Hua Tan, editors, *Trust and Deception in Virtual Societies*, pages 139–156. Kluwer Academic Publishers, 2001.

[19] B. Ramsdell. S/MIME version 3 message specification. Technical report, Internet Engineering Task Force, May 1998. Work in progress.

[20] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[21] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, MIT, 1996.

[22] Dawn Song and Jonathan Millen. Secure auctions in a publish/subscribe system. Available at http://www.csl.sri.com/users/millen/papers/dcca8.ps, 2000.

[23] Monique Calisti Stefan Poslad, Patricia Charlton. Specifying standard security mechanisms in multi-agent systems. In Rino Falcone, K. Suzanne Barber, Larry Korba, and Munindar P. Singh, editors, *Trust, Reputation, and Security: Theories and Practice, AAMAS 2002 International Workshop, Bologna, Italy, July 15, 2002, Selected and Invited Papers*, volume 2631 of *Lecture Notes in Computer Science*, pages 163–176. Springer, 2003.

[24] Stuart G. Stubblebine and Paul F. Syverson. Fair on-line auctions without special trusted parties. *Lecture Notes in Computer Science*, 1648:230–240, 1999.

[25] K. Suzuki and M. Yokoo. Secure combinatorial auctions by dynamic programming with polynomial secret sharing. In *Proceedings of Sixth International Financial Cryptography Conference (FC-02).*, 2002.

[26] OpenSSL Project team. OpenSSL. http://www.openssl.org/.

[27] H. Chi Wong and Katia P. Sycara. Adding security and trust to multiagent systems. *Applied Artificial Intelligence*, 14(9):927–941, 2000.

[28] Bennet Yee and Doug Tygar. Secure coprocessors in electronic commerce applications. In *Proceedings of The First USENIX Workshop on Electronic Commerce*, New York, New York, July 1995.

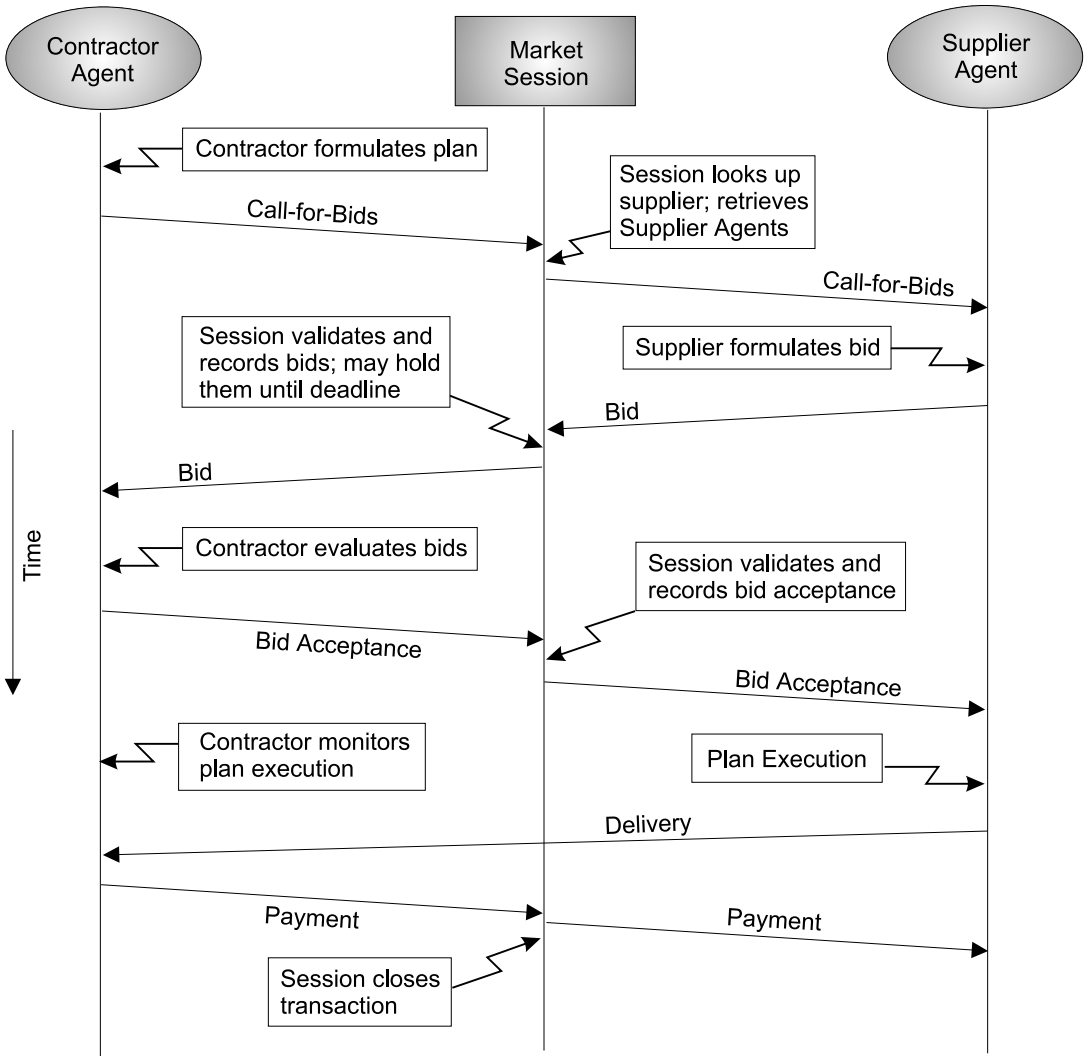

Fig. 1. MAGNET's original three step protocol

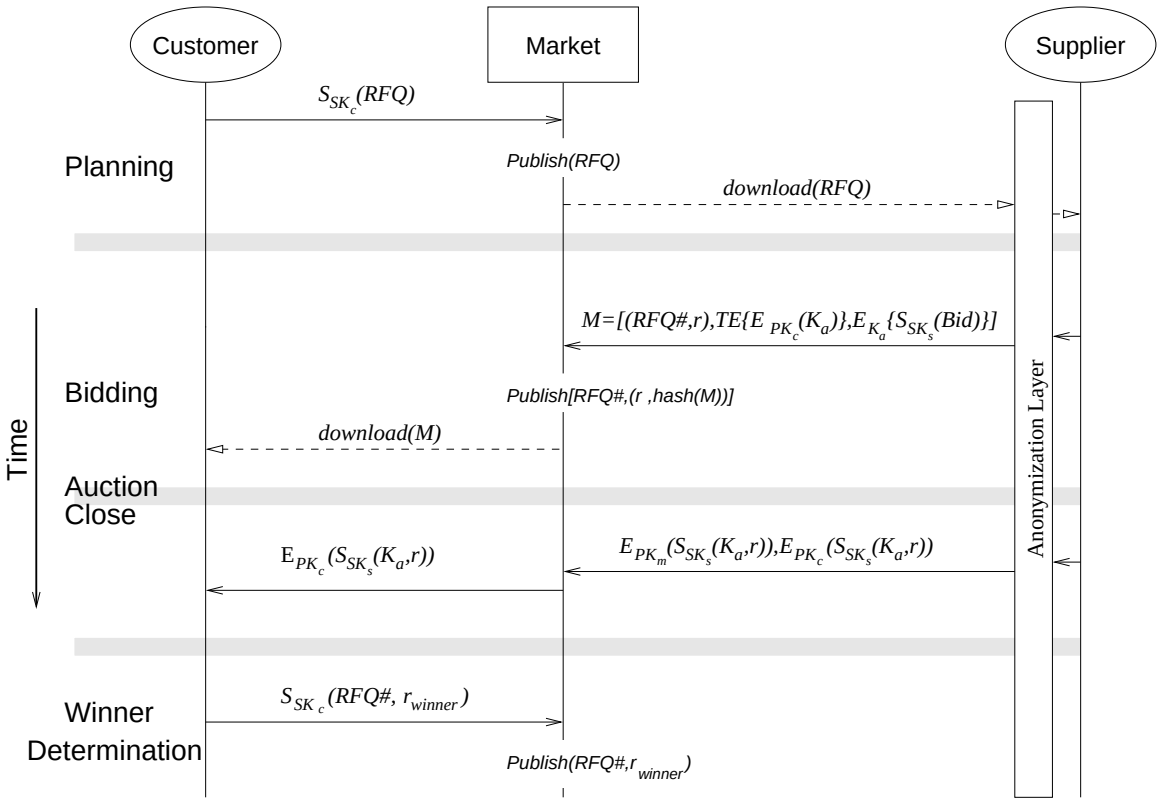

Fig. 2. The Secure MAGNET system

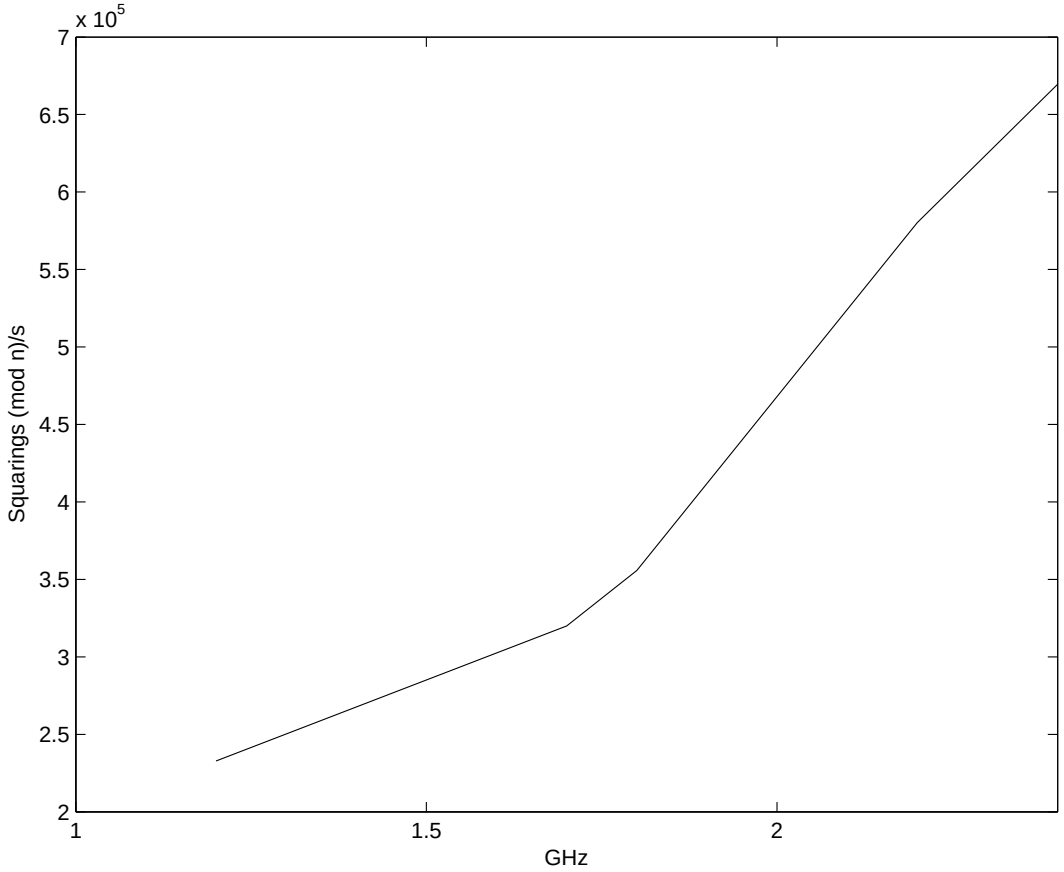

Fig. 3. Value of S for different processors

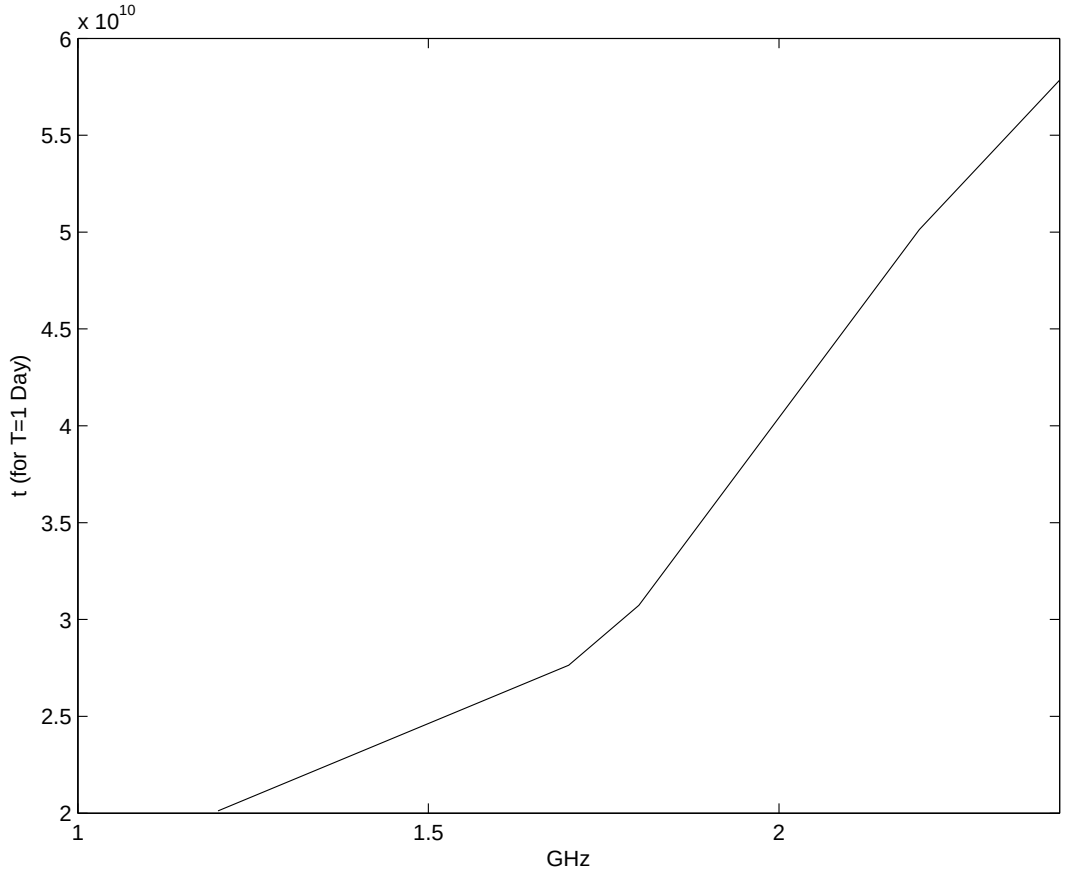

Fig. 4. Value of t (for T=1 day) for different processors