SAM: A Flexible and Secure Auction Architecture Using Trusted Hardware

Adrian Perrig[†] Sean Smith[‡]
Dawn Song[†] J. D. Tygar[†]

[†]UC Berkeley, [‡]Dartmouth College

{perrig,dawnsong,tygar@cs.berkeley.edu, sws@cs.dartmouth.edu} [*]

Abstract

Increasing numbers of economic transactions are conducted through on-line auctions. Nevertheless, most current auction implementations fail to address important security concerns. In particular, most auction systems force buyers and sellers to trust the auctioneer; alternative secure systems are in¤exible and have a high computational and/or communication overhead.

To overcome these limitations, we propose a secure auction marketplace (SAM) architecture, based on the recently available tool of high-performance, programmable secure coprocessors.

Unlike previous schemes, this approach provides a general framework that can incorporate arbitrary auction schemes by using different evaluation programs, as well as provide complex security properties by using the secure coprocessor and our auction protocols.

Our approach features strong security guarantees for the buyers and sellers without trusting the auctioneer, precise de£nition of the information disclosed during and after the auction, and high ¤exibility to adapt to new types of auctions.
Keywords: *Secure auction architecture, secure coprocessor.*

[*]Initial work was done while the £rst three authors were at IBM Research in summer, 1999. Later work, including contributions by the fourth author, were sponsered in part by the Defense Advanced Research Projects Agency under DARPA contract N6601-99-28913 (under supervision of the Space and Naval Warfare Systems Center San Diego), by the National Science foundation under grant FD99-79852, by the United States Postal Service under grant USPS 102592-01-Z-0236. The third author received additional support from the U.S. Department of Justice under contract 2000-DT-CX-K001, and from Internet2. Views and conclusions contained in this document are those of the authors and do not necessarily represent the of£cial opinion or policies, either expressed or implied of the US government or any of its agencies, DARPA, NSF, USPS, DOJ.

1 Introduction

This paper proposes the *Secure Auction Marketplace (SAM)*, an architecture for electronic auctions using trusted hardware. This architecture provides a way to ¤exibly and systematically address security, privacy, trust, and fraud problems — and is implementable with current off-the-shelf technology.

An *auction* is a general mechanism for commercial interaction. However, implementing auctions in the setting of distributed computing is complicated by several fundamental properties:

- Auctions involve *multiple parties*, such as the auctioneer, buyers, sellers — and possibly other stakeholders, such as government regulatory agencies.
- These parties have *con¤icting interests*.
- Auctions involve *private information*, such as bids, bidding strategies — and possibly templates for pattern-matching for likely fraud.
- Auctions involve *computation* on this information, such as execution of the auction, decisions on bids, recognition and suppression of fraud.

In a distributed setting, these properties create a fundamental trust challenge: we need to distribute this information and computation *among the parties themselves*, in way such that the computation is still correct, and all involved parties can still trust that their respective interests are preserved.

Recent advances in *secure coprocessing* provide a foundation to address these problems. COTS secure coprocessor platforms now provide:

- a *computing environment* which its owner can con£gure to carry out some arbitrary computation,

- and a *security environment* that enables remote parties to *authenticate* what this computation is, and to *trust* that this computation will proceed without observation or further manipulation by the owner of this device.

In this paper, we use this foundation to build our Secure Auction Marketplaces: havens that individual auctioneers can con£gure to carry out this auction computation, that resolve these trust issues in a much more general and ¤exible way than was possible with previous cryptographic approaches.

Section 2 provides more discussions of our auction model. Section 3 reviews previous approaches to this problem, and presents the secure co-processing technology that enables our approach. Section 4 presents our marketplace architecture. Section 5 demonstrates its value by discussing some of the security and privacy properties it achieves. Section 6 demonstrates its ¤exibility by presenting some avenues to extend this basic architecture. Section 7 concludes with some avenues for future work.

2 Auctions

The *auction* is an important economic mechanism which is widely used to sell a variety of commodities, such as treasury bills, mineral rights including oil £elds, real estate, art works, etc. With the growing popularity of the Internet, many traditional auctions are transforming into electronic auctions, and many new electronic auctions are being created. As a result, a number of web-based auction markets[1] have emerged. These range from relatively public markets such as auctions run by e-Bay, Amazon, and Yahoo! to Business-to-Business auctions (freemarkets.com, commerceone.com) to double auctions such as on-line stock markets. Compared to traditional auctions, electronic auctions have the advantages; they are global in scope, may be less expensive than traditional auctions. Participants in electronic auctions require neither physical presence nor (for *off-line* auctions) a connected electronic presence.

Our Auction Model. We consider general auctions. We view an auction as a triplet of trading rules, participant strategies, and result disclosure rules. The trading rules are announced before the auction. The sellers and bidders submit their strategies during the submission time as speci£ed in the trading rules. Then, during the auction evaluation time, the seller and bidder strategies are evaluated according to the trading rules and the result is determined. The result disclosure rules furthermore de£ne how the result is announced, i.e. public or only to the winner and the seller.

This model embraces a wide variety of auctions. See [7] for more information on auction types and variations. We brie¤y present the most common auctions:

[1] E.g., see www.auctioninsider.com/every.html.

- *Ascending-bid* auction (aka English auction): The bidders alternatively raise their bids or retire until only one bidder is left.

- *Descending-bid* auction (aka Dutch auction): The price continuously decreases until a bidder claims the good at that price.

- *First-price sealed-bid* auction: Each bidder submits her bid secretly, the highest bidder wins and pays the value of the bid.

- *Second-price sealed-bid* auction (aka Vickrey auction): Each bidder submits her bid secretly, the highest bidder wins and pays the value of the second-highest bid.

Our auction model also embraces more elaborate auctions, as shown in the following examples. Usually, auctions have one seller and many buyers, but in *reverse auctions* such as Priceline, only one buyer initiates the auction and many sellers place their offers. In the case of multiple buyers and sellers, the auction is called a *double auction*. If the resource sold in the auction is an on-going resource such as electricity, and the bids are cleared in short time intervals, the auction is known as a *continuous auction*. For example the stock market is an instance of a continuous double auction. If the bids contain multiple attributes such as volume and price, the auction is called a *multi-attribute auction*. For example, a future speci£es a contract to buy/sell a stock at a particular price at a certain time in the future, hence it is a multi-attribute auction. The bids can include conditions of other commodities, i.e. "I buy A at a price X only if I can buy B at a price Y.". Such an auction is called *combinatorial auction*.

Fraud. Buyers and sellers have direct interests in the outcome of the auction. However, often society itself has a broader interest in ensuring that certain kinds of fraudulent or criminal behavior does not occur. Our model can also extend to include these interests, by including attempts at fraud monitoring and suppression as additional trading rules.

Problem Statement. Electronic auctions are promising, but they also create new security challenges. The Internet bears many security threats; auctioneers, sellers, and bidders may all have more opportunities to cheat in an electronic auction. Many different types of auctions exist and they have various security requirements. The security properties we consider include:

- Authentication, privacy and anonymity of participants

- secrecy of bids and strategies

- controllable revelation of information about the auction including the £nal result

- the atomicity of the goods and charge (i.e., a winner only needs to pay if he gets the goods or vice versa) [21, 2].

- authentication by participants and stakeholders that any given auction actually followed the above rules.

Prior to our work, no solution satis£ed this wide variety of security requirements.

A main disadvantage of most of the existing auction schemes is that they require the sellers and bidders to trust the auctioneer. But in fact a common attack is that the auctioneer uses the seller and bidder information as an advantage to increase pro£ts.

A critical part of auctions and markets is the ability of participants to hide private information, such as their strategies, resources, and even identities. However, this privacy enables many types of fraud, such as:

- *Options front-running*, where a broker places a small order for himself before placing a large order for a client (which changes the price)

- *Money laundering in futures markets*, where a broker places symmetric orders, and later attributes the winning one to Alice and the losing one to Bob (thus enabling Bob to pay an untraceable bribe to Alice)

- *Fraudulent bids that win auctions* [22]. [2]

- *Collusion* among participants in an auction

- *Shills,* where the auctioneer driving up the price via fake bidders (perhaps to increase his commission), or otherwise exploiting the anonymity of bidders.

As marketplaces become increasingly distributed and automated, these problems become even worse. The problem is further complicated by mutual distrust among auctioneers, who would not want to share any of their secret information. When a critical infrastructure (e.g., the power-grid) is involved, the consequences of this rogue activity may change from simple fraud to a signi£cant national emergency.

3 Previous Work

Section 3.1 presents previous work on secure auctions. Section 3.2 presents the secure co-processing technology that enables our solution.

[2]There is also a recent case in which a child in Philadelphia placed fraudulent bids in electronic auctions, and won the bid [3].

3.1 Secure Auctions

Franklin and Reiter [4], and Harkavy, Tygar and Kikuchi [5], address auction security issues and they propose to use multiple auctioneers and variations of secret sharing schemes to reduce the trust on a single auctioneer[3]. But the approaches using multiple auctioneers are in general expensive, and dif£cult to adapt to other auction types and privacy requirements.

Stubblebine and Syverson [20] propose to use certi£ed mail and online notary services to reduce the trust on a single auctioneer in English auctions. This approach has the drawback that it requires to trust the additional services.

Naor, Pinkas, and Reingold present a scheme which is based on garbled circuits and oblivious transfer [11]. Their system can achieve similar security properties as SAM at the cost of a substantial communication and computation overhead. Although their approach is theoretically interesting, the large overhead prohibits a practical implementation.

3.2 Secure Coprocessors

Loosely de£ned, a *secure coprocessor* is a general-purpose computing environment that can be trusted to carry out its computation unmolested, even if an adversary has direct physical access to the device.

White, Comerford, and Weingart developed a high-end secure coprocessor prototypes [23, 24] for use in piracy suppression. Tygar and Yee [27, 26] used these prototypes to demonstrate the usefulness of secure coprocessors in distributed commerce applications. Smith and Weingart [18] then developed and implemented a logical and physical security architecture that enables a vendor to ship a generic secure coprocessor platform, that distributed application vendors can con£gure and maintain—while providing the core requirement that coprocessor applications can always prove "they're the real thing, doing the right thing," and also while accommodating the realities of trust issues and security ¤aws in complex software. This architecture was independently validated at FIPS 140-1 Level 4 [17], and is the basis for a COTS family of devices such as IBM 4758.

For our purposes, secure coprocessors provide three key features:

- that individual parties can install application code into their coprocessors;

- that, once installed, the application can proceed untampered—even by that party, who might advance his or her interests by tampering with the computation or observing its secrets;

[3]Note that Franklin and Reiter provide a powerful tool to address this problem - they have a system of e-cash where payment is provided automatically to the winner of an auction [4].

- that this untampered application can authenticate itself as such to remote participants.

This latter *outbound authentication* feature [14] is critical for us. In the IBM technology, devices leave the factory possessing a certi£ed key-pair, whose private key is con£ned (by hardware) to the security con£guration code that runs at boot time. This code generates and certi£es key-pairs for use by higher-level code in the device, and stores all private keys in tamper-protected memory. Application code can thus access private-key operations whose public keys are supported by a trust chain binding that key-pair to that application, in that software con£guration, on that untampered device. The literature (e.g., [6, 15]) provides more detail on how application development and deployment might work in practice.

4 SAM

Our Secure Auction Marketplace allows an untrustworthy auctioneer to conduct a wide variety of auctions ef£ciently.

The main problem we address is trust: in our design, the bidders do not need to trust the auctioneer, but they can get very strong security guarantees, such as:

- correct bidding result
- freedom from abuse
- con£dentiality of bids
- anonymity.

4.1 Installation and Deployment

Secure hardware alone cannot provide these properties: for example, if the auctioneer writes the software which runs on the secure coprocessor, a malicious auctioneer could simply write auction software which decodes the secret bids inside the secure coprocessor and sends it to the auctioneer in clear text. Achieving these properties requires that bidders and other participants be able to authenticate *which software* will conduct the auction. This is why we base our solution on an advanced secure coprocessor environment such as the IBM 4758-2 with outbound authentication, as discussed earlier.

The basic idea is that trusted software is loaded onto the secure coprocessor securely. The trusted software and the secure coprocessor together act as a *secure auction marketplace (SAM)*. This SAM then becomes an authenticated computational entity, whose internal state and operations cannot be examined or altered by an adversary—even one with direct physical access to that hardware.

In our case, this "trusted software" would be a secure auction operating system (SAOS). We sketch a potential API:

- register_auction(auction source code, auction spec, seller spec): auction advertisement
- register_bid(bid): bid con£rmation
- auction_status( ): status information

A party (such as ourselves) would obtain an application-developer certi£cate from the coprocessor manufacturer, and publish full information (source and signed executable) so that parties could both verify that this source matches this executable, and then install this executable in a virgin coprocessor.

This SAM could then use the coprocessor's outbound authentication API to obtain a pair of public/private keypairs (one for encryption, one for signatures) certi£ed to belong to that SAM. These give SAM the ability to provide authorized advertisement before the auction opening time by publishing the signed *auction advertisement*. The auction advertisement contains an unique auction ID, and an auction speci£cation which speci£es the trading rules. During the auction, the bidders send their bids (or strategy programs) encrypted with SAM's public key to the marketplace which returns a signed receipt. By binding the bid or strategy to an auction ID, the bidder is assured that the bid is evaluated according to the auction speci£cation, i.e. it is only evaluated after the auction closing time, and it is only evaluated for the intended auction by the speci£ed trading rules. After the auction closing time, the SAM evaluates the bids according to the published auction speci£cation, and outputs an authenticated result of the auction, in conformance with the result disclosure rules.

4.2 An Example of SAM

In this section, we describe a simple example of SAM. (We discuss various generalizations in Section 6.)

4.2.1 System Description

The SAM is based on a secure coprocessor which has the following software components: an auction controller and a bids collector, as shown in Figure 1. The auctioneer possesses such a SAM. The SAM has its unique public/private key pairs (i.e. one pair is used for encryption, and another pair for digital signatures) certi£ed through a CA. The private keys are generated inside the secure coprocessor and are hence not disclosed to any one else, including the auctioneer. To hold an auction, the auctioneer provides the auction speci£cation and publishes the auction advertisement. Bids are sent to SAM and evaluated in SAM. Finally, the result is computed and output by SAM.

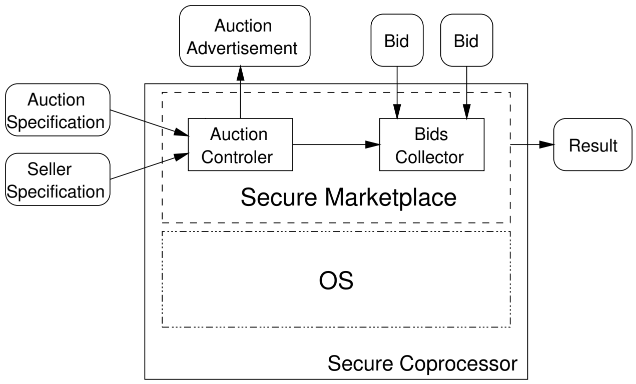

Figure 1. A Simple Secure Auction Marketplace

4.2.2 Phase description

This section describes the process of an auction phase by phase.

1. *Advertising Phase:* The auction speci£cation is input to the auction controller. The auction speci£cation includes the *opening time, closing time, auction evaluation algorithm, result disclosure rules*, and other parameters if necessary. The auction speci£cation is depicted in Figure 2. The auction controller generates a nonce *auction ID* and signs the concatenation of the auction ID and the auction speci£cation. An advertisement of the auction is then published. The advertisement should contain suf£cient information for participants to identify and evaluate the trustworthiness and behavior of the marketplace with the given parameters. Figure 3 shows the advertising phase.

2. *Opening Phase:* At the opening time of an auction, the auction controller enables the bids collector to collect incoming bids.

3. *Collection Phase:* During the bids collection phase, the bids collector receives bids. It validates the bids, including the veri£cation of the auction ID. If the check is successful, the bids collector signs the hash of the bid message with the auction ID and returns the signature. Furthermore, it inserts the bid into the bids table. A bid includes the auction ID, a bid strategy and other payload according to the speci£c auction, such as bidder ID and digital cash. The messages included in this phase are illustrated as the following:

   Bidder $\rightarrow$ Marketplace :
   $\{\text{auction ID}, bid, [\text{Optional Fields}]\}_{K_{SMP}}$

   Marketplace $\rightarrow$ Bidder :
   $\{\text{auction ID}, \text{Timestamp}, H(\text{bid message})\}_{K_{SMP}^{-1}}$

   Figure 4 depicts the opening and collection phase.

4. *Closing Phase:* At the closing time of an auction, the auction controller disables the bids collector to collect any more bids.

5. *Evaluation Phase:* The auction controller invokes the auction evaluation algorithm with the bids table as argument which generates the result. Figure 5 depicts the evaluation phase.

6. *Result-Disclosure Phase:* The auction controller signs the concatenation of the £nal result and the auction ID and sends it out.

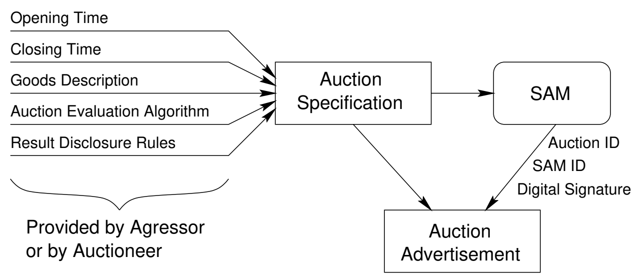

Figure 2. Auction speci£cation and auction advertisement. For this and the following three £gures: The round boxes represent entities, and the square boxes represent information.

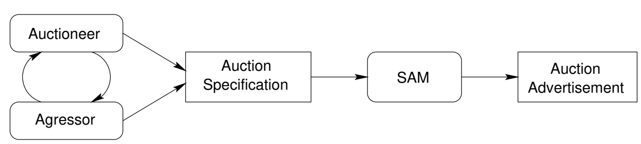

Figure 3. Auction advertisement phase

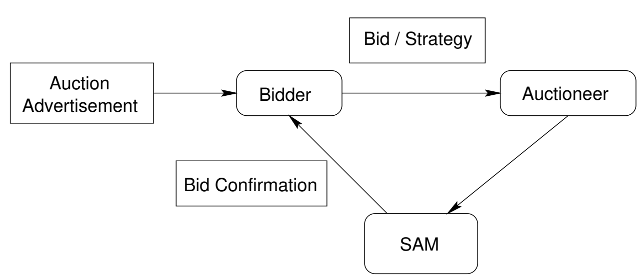

Figure 4. Auction opening and collecting phase

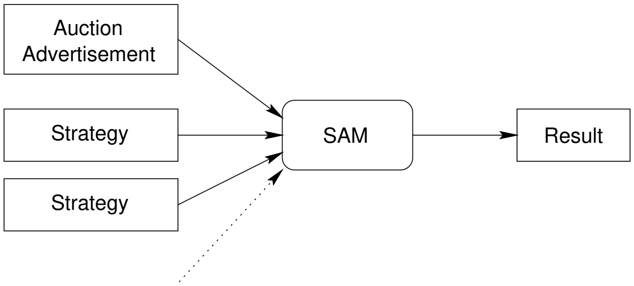

Figure 5. Auction evaluation phase

5 Properties

Our work features the following novel advantages:

The participants do not need to trust the auctioneer. In fact, our approach provides a high degree of secrecy and privacy for both sellers and bidders so that even the auctioneer cannot gain any more information than an outside observer. Hence, it automatically prevents fraud which relies on the auctioneer's insider knowledge.

This approach provides a universal framework for secure electronic auctions. Previous approaches have to use complex cryptographic algorithms and complicated system setups to achieve varying security properties. Our approach is more general and flexible to adapt to new auction schemes, and to provide different security properties. (We would like to emphasize that the cryptographic solutions currently fail to solve some basic auction problems with sufficient security.)

In SAM, the auction specification precisely states which information is private, public, or should be disclosed only to a subset of the participants. The auction result is disclosed in conformance with the result disclosure rules. This flexibility and ability of controllable information distribution is unachieved by the current cryptographic solutions for secure auctions.

This approach provides a universal framework for distributed electronic auctions. When a SAM is being used to continuously buy/sell an online resource, such as network bandwidth or electricity, we can not only distribute the devices into the network where this resource resides (which might be in an untrusted and dangerous physical location), but use the devices to directly execute the result of the deal.

When a SAM is used for off-line auction, it greatly reduces I/O bandwidth and eliminates the delay for the round-trip time for feedback and bid submission.

This approach provides a universal framework for fraud suppression in electronic auctions. As noted earlier, the SAM approach can combat fraud and other risks by providing a trusted place to calculate predicates on otherwise private participant information. For example:

- To prevent options front-running, the SAM can itself monitor for such correlations (without otherwise leaking bidder identity).
- To prevent money-laundering frauds, the SAM can require brokers to cryptographically commit to the identity of the end-client in each bid, at the time of the bid.
- To prevent false bids, the SAM can require some evidence of ability to pay before accepting a bid.
- To prevent rogue players from bringing down a critical infrastructure, the SAM can require some evidence of "ability to provide service" before accepting a seller.
- In an electronic futures market, the SAM also provides a nice place to clear orders, such as a contract-to-buy and a matching contract-to-sell.

In distributed markets (like the power grid example), these predicates can be calculated by distributed SAMs—although specifying the predicates and determining how to calculate them becomes much more interesting. Much work can be done here, depending on on the fraud and risk scenarios involved.

Trust Model We assume that the bidder does not trust the auctioneer; the seller does not trust the auctioneer; and the bidder and seller do not trust each other.

The correctness and the security of the auctions are only rely on the following factors:

- The secure coprocessor's tamper-resistance, code loading techniques, and outbound authentication API work correctly. (The FIPS validation helps us here.)
- The software components, auction controller and bids collector work correctly.

Secrecy of the strategies The strategies of the sellers and the bidders are encrypted with SAM's public key and reside in SAM after being decrypted. As a result, no one else can have access to these secret information.

Controllable disclosure of the final result The way in which the final result is published is specified in the auction specification and followed by SAM. As a result, no one else can get more information about the final result than what is specified in the auction specification.

When digital cash, e-check or credit card number are included in the bids for payment, the secret information is kept inside SAM. The losers' information is not revealed to anybody else, while the winner's information is only revealed to related party for collecting the money.

Privacy and Anonymity: The identity and private information of sellers and bidders are kept secret. The secret used for authorization of sellers and bidders are also kept secret.

Information Fairness: The information about the auction is published in the advertisement and cannot be modi£ed by any one including the auctioneer (because the advertisement is signed by SAM). No one gets more information about others than any one else. How the £nal result is to be distributed is also speci£ed in the auction speci£cation. Hence it is fair in the sense that no one can get more information than others without being speci£ed publicly.

Time Fairness We assume that SAM has an internal clock with a small drift rate and cannot be altered without authorization. The sellers, bidders and processors within the SAM have a way to synchronize their clocks with each other. The SAM controls the opening time and closing time for the auction. Hence, everybody experiences the same opening time and closing time.

Furthermore, on-line English or Dutch auction usually have the unfairness that different bidders experience different network delay time. By using SAM, we can eliminate this unfairness. Bidders can submit their strategies into SAM during the bidding time and then, the strategies are evaluated using the English or Dutch auction fashion. Because the strategies are already inside SAM, no network delay time interferes during evaluation time.

Moreover, some major exchanges tried to address time fairness problems by shipping announcements encrypted to trusted machines at geographically distributed sites; these machines then ensure that announcements are publicly revealed at the same real time. We can adopt and extend this technique in distributed SAM marketplaces, where appropriate.

Opportunity Fairness A receipt is generated for every bid that is received by SAM. If a bidder gets the receipt, it is guaranteed that its bid will be evaluated in the evaluation phase, should the auction run to completion. Should the auction fail (e.g., because the SAM hardware experienced a power failure or other catastrophe, perhaps caused by a malicious auctioneer), SAM will handle the failed auction in accordance with the policy indicated in its original spec. This policy can range from complete canceling to complete re-execution, and can include provisions to prevent the SAM from doing anything else until it has handled this failure.

6 Extensions

Our approach to the auction problem replaces special-purpose cryptographic protocols with generic crypto channels into programmable secure hardware. The existence of a trusted computational entity in this system permits many generalizations of this approach, to handle variations on evaluation algorithms, bidder strategies, seller strategies, and other aspects of the auction process.

In this section, we sketch some of these extensions.

6.1 Generalization on Evaluation Algorithms

How does the evaluation algorithm get installed in SAM in the £rst place? In the most straightforward implementation, we might simply hardcode it into the SAM software. However, the computational nature of SAM enables more ¤exible approaches. For example, the SAM can contain different types of auctions, such as English auction and Dutch auction. The speci£cation for any particular auction instance would include a parameter which speci£es the type of auction.

For even more ¤exibility, SAM could permit the evaluation algorithm, expressed in some suitable speci£cation language, to be included in the auction speci£cation, as another parameter to the the auction controller. SAM would use this speci£cation along with the seller's strategy to evaluate bids. With the increased popularity of double auctions and bundling auctions, and the continued emergence of new auction types, this extensibility would be an advantage.

As with all downloaded programs, the speci£cation language — and the SAM module that catches and uses it — needs to be able to protect SAM integrity from potentially malicious speci£cations.

The evaluation algorithm can be written in a speci£cation language which will then be interpreted by an interpreter in the auction controller. Alternatively, the evaluation algorithm can be in a real program such as Java bytecode or C program. In any case, the language that the evaluation algorithm is speci£ed in needs to be well restricted in the sense that it is guaranteed that a valid evaluation algorithm cannot do anything bad such as manipulate the bids table. Many solutions can be possible, such as simple sandboxing. Architectures such as Java 1.2 may provide an avenue to address this problem. We also need to make sure the correctness of the implementation of the speci£cation language in the sense that the implementation matches the speci£cation. Covert channels can also be an issue.

The language needs to be such that bidders can quickly make effective judgments about the nature of the auction. It is an interesting research question how to design this speci£cation language or a subset of an existing programming language such as Java or C to evaluate bids.

For such extensibility to be effective, the evaluation algorithm would be published before the auction starts as part of the auction advertisement. The signed auction advertisement needs to bind the evaluation algorithm to the auction. This binding can be achieved with many techniques, such as having the auction controller include a hash of the program, or even the full source, in the auction speci£cation.

6.2 Generalization on Bidding

Bids

Authorization of bidders is important in many cases, such as bidding for government contract and bidding for natural resources. In order to provide authorization of bidders, the bids collector can contain a an *authority checker* subcomponent which checks whether the bidder is authorized. Only authorized bids can then be inserted to the bids table. Several solutions for the authorization mechanism exist. For example, an authorized list can be established in the authority checker in which each entry is a pair (bidder ID, secret). If a bid contains the correct bidder ID and secret, it is authorized. An even simpler solution is to establish a shared group secret for all the authorized bidders. Any bid which contains the correct secret is regarded as authorized. Well-designed protocols can prevent these shared secrets from leaking out without the help of any authorized bidder. Schemes based on a PKI are also possible.

As another generalization, the bidders can include *digital cash* with their bids. At the end of the auction, the winner's digital cash can be collected as the payment; the others are canceled. Textbook atomic transaction techniques can be used to guarantee that the goods is given to the winner only if the money is transferred to the seller's account and the bidder does not need to pay anything if the goods is not given to him.

The bidder can include a *secret key* with their bids. At the end of the auction, the goods (if it is electronic) or some secret information about picking up the goods can be encrypted with the secret key and published. Hence, the winner can get the goods without revealing any information about himself and hence to be fully anonymous.

Non-repudiation of bids can also be a desired property. To achieve this, either the bidders can sign the bids, or they can include authentication secrets in the bids.

Without the SAM to carry out the market, including sensitive information such as digital cash and bidder-speci£ed secret keys in the bids raises various risks. For example, the auctioneer can then get the bidder-speci£ed secret keys and hence get the goods without paying for anything.

Bidding Strategies

The bids can also take more complicated format, for example, using a strategy speci£cation language which can be interpreted or even be a real executable program. The syntax and the semantics of the strategy speci£cation language are published. Similar to the issues previously discussed on evaluation speci£cation language, the strategy speci£cation language design bears the same security concerns as the evaluation algorithm speci£cation language design discussed in the previous subsection.

The bidder may also enter a meta-strategy that expresses their strategy for some sequence of auctions. These meta-strategies may include such features as: reselling something that was purchased, and exchanging information with other meta-strategy agents.

6.3 Generalization on Seller Strategies

In some auction schemes, a seller might need to enter some secret data into the secure marketplace. For example, a seller might want to specify a lower limit below which he does not want to sell the goods. Bidders are allowed to know that such a lower limit exists but cannot know what exactly the lower limit is. In this case, the seller can input its secret data encrypted with the SAM's public key which is dedicated to encryption, or establish a shared secret with SAM to encrypt its secret data. The seller strategy needs to be authenticated to make sure that it comes from the seller. One solution could be that the seller's public key is included in the advertisement of the auction and the seller signs its secret data with its private key before it is encrypted with the SAM's public key.

Similar to the strategy speci£cation language mentioned before, sellers can also specify their strategy or meta-strategy of selling in the strategy speci£cation language. And the same security concerns also apply here.

The seller may include a pledge (or actual digital cash) with the offer to sell; the advertised auction spec will indicate how that pledge is to be disposed of should various bad things (such as the seller not actually producing the item to be sold) occur.

6.4 Generalization on Commodity

For *electronic* commodities, the SAM can transfer ownership of an electronic commodity as an atomic part of the transaction. For some types of commodities (such as contracts, or signed things), the SAM can transfer ownership in such a way that the original owner need never be involved—and the object can be resold several more times without ever leaving the SAM network. For commodities that are electronically controllable, the SAM's coprocessor can itself directly execute the result of a transaction.

For some types of commodities, like encrypted network links, transferring data from one link controlled by Alice to one controlled by Bob would require a trusted party anyway. A SAM could execute the deal as well as performing this cryptographic transformation as a result of the deal.

6.5 Generalization on Auction Types

In the above section, we limited ourselves to the use of a single coprocessor. Such an approach has architectural elegance, but is also necessarily limited by the resources available to the coprocessor. How can we scale?

It is easy to imagine networking secure coprocessors, since auctions are often hierarchical, this is a natural step. An auction in a limited geographical area (a county) results in a single winner (or, in the case of multiple goods be auctioned, is n winners.) The winner(s) compete in a larger nationwide auction, and the winner(s) of that auction compete in an international auction. Auctions can not only take the form of single seller and multiple buyers, but also single buyer and multiple sellers, or multiple sellers and multiple buyers. All these types can very easily £t in the framework of SAM (although de£ning and ensuring fairness in these multiple-multiple settings could be interesting, should not all players £t inside a single coprocessor platform).

A particularly intriguing feature of SAM auctions is that they can provide for powerful time-fairness. With suf£ciently robust clock synchronization,[4] SAM can timestamp each bid. The £nal auctioneering secure coprocessor waits until the auction end time plus the maximum possible clock skew among the secure coprocessor plus the maximum possible latency in the network. Clearly, the £nal auctioneering coprocessor would receive all valid bids by this time. Because bids were timestamped by a secure source, only valid bids could be accepted. This has important implications for online stock markets.[5] With computer-assisted trading, it appears that, without secure timestamping, bidders who were on the antipode of the secure market would have bids delayed by at least one second - and typically far more because of transmission delays and speed of light considerations. But with timestamping we could release continuous auction information to bidders and sellers and also fairly consider bids and asks at the coprocessor by using the timestamp data.

[4] Smith, Johnson, and Tygar [16]; and Smith [19] contain relevant material on using secure coprocessors for clock synchronization.

[5] It is amusing to note that this same technique also could be used for many more applications: for example, the 2000 US election vote in Florida had many controversies. One was the question of inclusion of votes by military personnel (on ships, for example) who could not postmark their ballots. Secure coprocessor timestamping would help here.

So far, we have only considered auctions where bidding occurs in a well-de£ned interval; instead, the bidding could be *continuous.* Sellers and bidders send in their strategies into SAM. Information about currently available resources can be updated periodically, i.e. every second or every £ve minutes. Participants can also change their strategies if needed, for example, if a new demand pattern appears. SAM is essential for carrying out this type of auction, because £rst, strategies reside in SAM and hence can be evaluated ef£ciently and eliminate the round-trip time for publishing information and submitting bids as inevitable in traditional solution; second, SAM can physically be integrated at anywhere including malicious places and environment that is harmful for human being.

The SAM can ensure that auction specs enforce rules such as: a minimum number of bids must occur before an auction can be executed, or a receipt must be received from some number of authenticated advertising sources.

6.6 Fraud and Anomaly Detection

Electronic auctions are naturally at risk of fraud. The existence of a trusted computational entity in SAM provides some interesting potential for detecting and suppressing such fraud.

A SAM can execute anomaly or fraud detection algorithms on the data at an auction or sequence of auctions. The fact that SAM is already a trusted participant helps address what otherwise would be a privacy concern: how to do anomaly detection on data that is supposed to be private.

Networks of SAMs, even for different auctioneers, can jointly and possibly concurrently execute these algorithms. They can be used to detect wide-range fraud. For example, in critical resources such as electricity power, such SAM group can be used to detect when some single buyer wants to buy all the resources and then become a monopoly to gain pro£t. Some similar enforcement policy can be deployed to prevent fraud while preserve desired secrecy and privacy of legitimate participants. The auction spec can be constructed so that the existence (and perhaps trustworthiness, as witnessed by some appropriate party) of these algorithms is indicated, but the details of the algorithms remain secret.

7 Conclusions and Future Work

In this work, we have argued for the use of tamper-resistant hardware to build secure auction markets. This is a conceptual design – performance will be a key issue in any actual implementation. It is not only our architecture which impacts on performance, but because trusted hardware typically lags behind off-the-shelf commercial processors. We believe our architecture can be highly successful using existing trusted hardware platforms.

A particularly intriguing area for future research is the actual question of how we distribute work in the case of multiple simultaneous auctions (potentially with overlapping groups of bidders and overlapping groups of sellers.) Such a solution is complicated by the requirement that prices be allocated fairly. (While third party arbitrage could help ensure this, we suspect that it is possible to realize this directly in the mechanism design.)

Regardless of the details of implementation, trusted hardware can certainly provide a proven fair approach to realizing auctions. The advantages extend across a wide-range of auction types – from small, *ad hoc* auctions to large ongoing auction markets. Given the degree of fraud possible with existing auction architectures, there is a clear case for using trusted hardware to ensure fairness and security.

References

[1] *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, Apr. 1987. IEEE Computer Society Press.

[2] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In *Proc. 4th ACM Conference on Computer and Communications Security*, pages 6, 8–17, 1997.

[3] D. Edelson. 13-year-old makes 3m dollars in bids on ebay. The Risks Digest, April 30 1999.

[4] M. Franklin and M. Reiter. The design and implementation of a secure auction service. *IEEE Transactions on Software Engineering*, 22(5):302–312, May 1996.

[5] M. Harkavy, J. Tygar, and H. Kikuchi. Electronic auctions with private bids. In *Proceeding of 3rd USENIX Workshop on Electronic Commerce*, 1998.

[6] S. Jiang, S. Smith, and K. Minami. Securing web servers against insider attack. In *Annual Computer Security Applications Conference*. ACSA/ACM, December 2001. (to appear).

[7] P. Klemperer, editor. *The Economic Theory of Auctions (2 vols.)*. Edward Elgar Publishing, 2000.

[8] R. McAfee and J. McMillan. Auctions and bidding. *Journal of Economic Literature*, 1987.

[9] P. Milgrom. *Auction Theory*. Cambridge University Press, 1985.

[10] T. Mullen and M. Wellman. The auction manager: Market middleware for large-scale electronic commerce. In *Proceeding of 3rd USENIX Workshop on Electronic Commerce*, 1998.

[11] M. Naor, B. Pinkas, and O. Reingold. Privacy preserving auctions and mechanism design. In *Proc. 1st ACM Conference on Electronic Commerce*, October 1999.

[12] S. J. Rassenti, V. L. Smith, and R. L. Bul£n. A combinatorial auction mechanism for airport time slot allocation, 1982.

[13] P. Sanders, S. Rhodes, and A. Patel. Auctioning by satellite using trusted third party security services. *11th IFIP conference on information security*, 1995.

[14] S. Smith. Outbound authentication for programmable secure coprocessors. Technical Report TR2001-401, Department of Computer Science, Dartmouth College, March 2001. http://www.cs.dartmouth.edu/˜pkilab/oatr.pdf.

[15] S. Smith. Webalps: A survey of e-commerce security and privacy applications. *ACM SIGecom Notices*, 2:29–36, 2001.

[16] S. Smith, D. Johnson, and J. Tygar. Completely asynchronous optimistic recovery with minimal rollbacks. In *25th International Symposium on Fault-Tolerant Computing*, 1995.

[17] S. Smith, R. Perez, S. Weingart, and V. Austel. Validating a high-performance, programmable secure coprocessor. In *22nd National Information Systems Security Conference*, October 1999.

[18] S. Smith and S. Weingart. Building a high-performance, programmable secure coprocessor. *Computer Networks (Special Issue on Computer Network Security)*, 31:831–960, 1999.

[19] S. W. Smith. *Secure Distributed Time for Secure Distributed Protocols*. PhD thesis, Carnegie Mellon University, 1994.

[20] S. Stubblebine and P. Syverson. Fair on-line auctions without special trusted parties. In *Financial Cryptography'99*, 1999.

[21] J. D. Tygar. Atomicity in electronic commerce. In *Proc. 15th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 8–26, 1996.

[22] H. Varian. Mechanism design for computerized agents. In *Usenix Workshop on Electronic Commerce, July 11-12, 1995, New York*, New York, July 1995.

[23] S. Weingart. Physical security for the microabyss system. In *Proceedings of the IEEE Symposium on Research in Security and Privacy* [1].

[24] S. White and L. Comerford. Abyss: A trusted architecture for software protection. In *Proceedings of the IEEE Symposium on Research in Security and Privacy* [1].

[25] P. Wurman, W. Walsh, and M. Wellman. Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems*, to appear.

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

[27] B. S. Yee. *Using Secure Coprocessors*. PhD thesis, Carnegie Mellon University, 1994.