The Design and Implementation of a Secure Auction Service

Matthew K. Franklin Michael K. Reiter

AT&T Bell Laboratories, Holmdel, New Jersey, USA
franklin,reiter@research.att.com

Abstract

We present the design and implementation of a distributed service for performing sealed-bid auctions. This service provides an interface by which clients, or "bidders", can issue secret bids to the service for an advertised auction. Once the bidding period has ended, the auction service opens the bids, determines the winning bid, and provides the winning bidder with a ticket for claiming the item bid upon. Using novel cryptographic techniques, the service is constructed to provide strong protection for both the auction house and correct bidders, despite the malicious behavior of any number of bidders and even a constant fraction of the servers comprising the auction service. Specifically, it is guaranteed that (i) bids of correct bidders are not revealed until after the bidding period has ended, (ii) the auction house collects payment for the winning bid, (iii) losing bidders forfeit no money, and (iv) only the winning bidder can collect the item bid upon. We also discuss techniques to enable anonymous bidding.

1 Introduction

Technology has replaced many human procedures with electronic ones. Unfortunately, much of the tradition, culture, and law that has been developed to provide protection in human procedures cannot readily be adapted to afford the same protection in electronic procedures. The study of cryptographic protocols can be viewed as a technical response to this loss of more traditional means of protecting ourselves. Indeed, Diffie has argued that communication security is "the transplantation of fundamental social mechanisms from the world of face to face meetings and pen and ink communication into a world of electronic mail, video conferences, electronic funds transfers, electronic data interchange, and, in the not too distant future, digital money and electronic voting" [8].

As this statement hints, one human procedure whose protections are threatened by electronic advances is commerce. While many proposals have been put forward to guide the transition to electronic commerce (e.g., [4, 19, 2]), most of these proposals provide for only simple transactions involving little negotiation or competition among buyers and sellers. In contrast, many financial vehicles, such as auctions, exchanges, and general markets, do not conform to this simplistic view of commerce. We believe that the transition to electronic commerce should not preclude such vehicles, but rather should make them more accessible.

We have begun an effort to examine some of these financial vehicles to understand what is required to adequately implement them in electronic systems. In this paper we present an approach to implement one such vehicle, namely sealed-bid auctions. A sealed-bid auction is one in which secret bids are issued for an advertised item, and once the bidding period closes, the bids are opened and the winner is determined according to some publicly known rule (e.g., the highest bidder wins). Sealed-bid auctions are used, for example, in the auctioning of mineral rights to U.S. government-owned land, in the sale of artwork and real estate, and in the auctioning of government procurement contracts [18].

Our study of sealed-bid auctions is motivated not only by their practical importance, but also by the novel security problems that they pose. First, central to the fairness of a sealed-bid auction is the secrecy of sealed bids prior to the close of the bidding period. That is, the *timing* of the disclosure of bids is crucial. Second, auctions require nonrepudiation mechanisms to ensure that payment can be collected from winning bidders—as evidenced by the fact that in a recent FCC auction of interactive video and data service licenses, 13 winning bidders defaulted on their bids, forcing a second auction to be held [12]. Third, due to secrecy requirements surrounding sealed-bid auctions, it may be difficult for outsiders to have confidence in the validity of the auction. Fourth, some types of sealed-bid

1081-6011/95 $04.00

auctions should enable bidders to remain anonymous. These problems are only exacerbated when one considers the implementation of auctions in distributed computer systems, or the possibility of a corrupt agent in the auction house collaborating with bidders.

In this paper we present a secure distributed auction service that supports the submission of monetary bids for an auction and ensures the validity of the outcome, despite the malicious collaboration of arbitrarily many bidders and even a constant fraction (e.g., one-third) of the auction servers comprising the service. Our auction service addresses all of the security issues mentioned above. In particular, the auction service is guaranteed to declare the proper winning bidder, and to collect payment in the form of digital cash from only that bidder. It is guaranteed that no bid is revealed prior to the close of the bidding period. Moreover, it is possible for bidders to submit anonymous bids. Our focus in this work is on an efficient and practical approach to performing auctions, and we have implemented a prototype of our service to demonstrate its feasibility. The performance of this implementation indicates that our approach to performing auctions is feasible using off-the-shelf workstations for auction servers, even for large auctions involving hundreds of bids.

More generally, our auction service demonstrates novel techniques for handling and protecting electronic currency in competitive environments, using both old and new cryptographic methods. We are working to extend these techniques to address issues in other competitive financial vehicles, including more general forms of auctions, markets, and electronic gaming.

The rest of this paper is organized as follows. In Section 2 we describe the security policy that should govern a sealed-bid auction. In Section 3 we give preliminary definitions that will be used in the paper. In Section 4 we describe a new cryptographic primitive called *verifiable signature sharing*, which is an important enabler for the efficient implementation of secure auctions. We present our auction protocol in Section 5, and discuss its security and performance in Sections 6 and 7, respectively. We present extensions to our protocol in Section 8 and conclude in Section 9.

2 Secure auctions

Informally, a sealed-bid auction consists of two phases of execution. The first is a *bidding period*, during which arbitrarily many bidders can submit arbitrarily many sealed bids to the auction. At some point the bidding period is *closed*, thus initiating the second phase in which the bids are opened and the winner is determined and possibly announced. In general, the rule by which the winner is determined can be any publicly known, deterministic rule. When convenient, however, we assume that this rule dictates that the highest bidder be chosen the winner.

As mentioned in Section 1, there are numerous possibilities for corruption and misbehavior in a sealed-bid auction. Possibly the most difficult to counter are those that involve the misbehavior of *agents* in charge of executing and overseeing the auction (e.g., employees of the auction house), especially when this behavior involves collaboration with certain bidders. Below are several examples of behavior that could yield an improper auction, many of which may be very feasible in a naive electronic implementation of auctions.

- Prior to the close of the bidding period, an agent of the auction house opens submitted bids and informs a collaborator of their amounts (so the collaborator can submit a bid for the minimum amount needed to win the auction).
- An agent manipulates the closing time of the bidding period. For example, an agent attempts to prematurely close the bidding period in an effort to exclude some bids.
- After the close of the bidding period, a bidder arranges to withdraw a bid or insert a bid, in collaboration with an agent of the auction house.
- An agent awards the auction item to someone other than the winning bidder (and goes undetected because bids are not made public).
- An agent of the auction house collects payment from losing bidders (e.g., by informing each that it won), or collects payment from the winning bidder but fails to provide the means for that bidder to obtain the item bid upon.
- The winning bidder refuses to pay the auction house (e.g., by disclaiming the bid or claiming that it lacks sufficient funds).

It is worth noting that in a naive electronic implementation of a sealed-bid auction, some of the above problems could arise simply due to the benign failure of the auction service or a bidding process. For example, the next-to-last problem could arise if the auction service is not fault-tolerant, collects money from the winning bidder, and then fails before granting the item to the bidder. Similarly, the last problem could arise if a bidding process submits a bid and then fails.

Our auction service prevents the above behaviors and most other "attacks" on auctions of which we are aware, despite the malicious behavior of arbitrarily many bidders and even a constant fraction of the auction servers comprising the service.

We describe the properties provided by our auction service in two categories, namely Validity properties and Secrecy properties. Below and throughout this paper, a process (bidder, server, etc.) is said to be *correct* if it always follows the specified protocols. A *faulty* process, however, may deviate from the specified protocols in any fashion whatsoever; i.e., "Byzantine" failures are allowed.

Validity

1. The bidding period eventually closes, but only after a correct auction server decides that it should be closed.

2. There is at most one winning bid per auction, dictated by the (deterministic) publicly-known rule applied to the well-formed bids received before the end of the bidding period.

3. The auction service collects payment from the winning bidder equal to the amount of the winning bid.

4. Correct losing bidders forfeit no money.

5. Only the winning bidder can collect the item bid upon.

Secrecy

1. The identity of a correct bidder and the amount of its bid are not revealed to any party until after the bidding period is closed.

In addition, our auction protocol can be modified to allow for the submission of anonymous bids.

One class of attacks that our auction service does *not* address are those that involve collaboration among bidders to "fix" the price that wins the auction. For example, bidders could collude to bid no more than a certain amount. We also do not address attacks in which messages to and from bidders are intercepted, delayed, or otherwise manipulated in transit. For example, we do not guarantee that a bid submitted by a correct bidder will be included in the auction (although it will be if it is received intact before the close of the bidding period). We emphasize, however, that the attacks discussed in this paragraph have no effect on the Validity or Secrecy properties described above.

3 Preliminaries

In this section we review some primitives that are used in our auction protocol. The following notation will be used in the remainder of the paper. The encryption of m with a public key K is denoted $\langle m \rangle_K$, and the decryption of m with private key K^{-1} is denoted $(m)^{K^{-1}}$. The digital signature of a message m by a process P (i.e., with P's private key) is denoted $\sigma_P(m)$. We will introduce additional notation in the following sections as necessary.

3.1 Group multicast

Group multicast is a class of interprocess communication primitives by which messages can be multicast to a group $\mathcal{G}$ of processes. Our auction service employs three types of group multicast primitives, namely unreliable, reliable, and atomic. Each of these multicast primitives enables a process $S \in \mathcal{G}$ to multicast a message to the members of $\mathcal{G}$.

The weakest of these multicast primitives is unreliable multicast. We denote the unreliable multicast of a message m from a process $S \in \mathcal{G}$ to the group $\mathcal{G}$ by

$$S \rightarrow \mathcal{G}: \quad m$$

Unreliable multicast provides the property that if S is correct, then all members of $\mathcal{G}$ receive the same sequence of unreliable multicasts from S, which is the sequence of unreliable multicasts initiated by S. In particular, unreliable multicasts are authenticated and protect the integrity of communication. However, no guarantees are made regarding unreliable multicasts from a faulty S.

The second multicast primitive is called reliable multicast, also known as Byzantine agreement [17]. We denote the reliable multicast of message m from a process $S \in \mathcal{G}$ to the group $\mathcal{G}$ by

$$S \xrightarrow{R} \mathcal{G}: \quad m$$

Reliable multicast provides all of the properties of unreliable multicast. In addition, it strengthens these properties by ensuring that for each $S \in \mathcal{G}$, all correct members of $\mathcal{G}$ receive the same sequence of reliable multicasts from S, regardless of whether S is correct or faulty. However, reliable multicasts from different members can be received in different orders at each member of $\mathcal{G}$.

The third and strongest multicast primitive is atomic multicast. We denote the atomic multicast of message m from a process $S \in \mathcal{G}$ to the group $\mathcal{G}$ by

$$S \xrightarrow{A} \mathcal{G}: \quad m$$

Atomic multicast provides all of the guarantees of reliable multicast, and strengthens them by ensuring that all correct members of $\mathcal{G}$ receive the same sequence of atomic multicasts (regardless of their senders).

Because processes executing our auction protocol must sometimes block awaiting the receipt of reliable or atomic multicasts, it is necessary to provide some degree of failure detection to guarantee progress in the case that a faulty member does not multicast a message on which others are waiting. Moreover, correct group members must concur on the set of messages multicast by such a member prior to its failure. The reliable and atomic multicast protocols that we have implemented provide these properties [21].

In addition to multicasts from within a process group, our auction protocol also requires the ability for any arbitrary process $B \notin \mathcal{G}$ to atomically multicast messages to $\mathcal{G}$. We denote such a multicast of a message m by

$$B \xrightarrow{A} \mathcal{G}: \quad m$$

Atomic multicasts from outside the group are provided the same total ordering guarantee as those from within the group. That is, all correct members of $\mathcal{G}$ receive the same sequence of atomic multicasts, regardless of the origin of those multicasts. However, unlike atomic multicasts from within the group, atomic multicasts from outside the group are not authenticated, but rather are anonymous (i.e., they do not indicate their senders). Moreover, failure detection of processes outside the group is not provided.

The multicast protocols that we have implemented can tolerate the failure of t members of a group of size n (and any number of non-member failures) provided that $n \geq 3t + 1$ [21].[1] As described in Section 4, however, this is not the limiting factor in the fault-tolerance of our auction protocol.

3.2 Threshold secret sharing schemes

A (t, n)-threshold secret sharing scheme [1, 25] is, informally, a method of breaking a secret s into n *shares* $sh_1(s), \ldots, sh_n(s)$, so that $t + 1$ shares are sufficient to reconstruct s but t or fewer shares yield no information about s. In this paper, we make use of the polynomial based secret sharing scheme due to Shamir [25]. In this scheme, the secret s is an element of a finite field F and the i-th share is $sh_i(s) = f(i)$, where $f(x)$ is a degree t polynomial such that $f(0) = s$ and such that the other coefficients are chosen uniformly at random from F. Interpolation of any $t + 1$ shares reconstructs $f(x)$ and hence the secret s. F is typically taken to be the integers modulo p for some prime p larger than the secret. This scheme works for any threshold t, $1 \leq t < n$.

As observed by Feldman [13], if the results obtained by applying a public one-way function to each share are known, a process attempting to reconstruct the secret can verify that a share has not been altered prior to using it in reconstruction. In this way, the alteration of up to $n - t - 1$ shares can be tolerated. Our auction protocol will make use of this observation.

3.3 Electronic money

In its basic form, an electronic money or "digital cash" scheme [4] is a set of cryptographic protocols for (i) a customer to withdraw electronic money from a bank, (ii) the customer to use the money to purchase something from a vendor, and (iii) the vendor to deposit the money in its account with the bank. These protocols protect the security interests of the parties involved, by ensuring that the customer's identity cannot be linked to the purchase (i.e., anonymity), that each party accepts only valid electronic money, and that the customer cannot undetectably reuse or forge money. For the purposes of this paper, we will not consider cash schemes that require physical assumptions (e.g., tamper-proof smart cards) [11].

A money scheme is said to be "off-line" [5] if the purchase protocol does not involve the bank; otherwise the scheme is said to be "on-line." In a typical on-line scheme, the vendor queries the bank to determine whether the "coin" that a customer is attempting to use in a purchase has already been spent. In an off-line scheme, the bank is not consulted during purchases, and hence reuse cannot be prevented. However, the customer's identity can be embedded in each coin in a way that is accessible if and only if the same coin is used for more than one purchase. When the copies are eventually deposited, the bank will learn the identity of the reuser. In this paper, we consider only off-line cash schemes.

The auction protocol that we present in this paper will work with most off-line cash schemes. For this reason, in stating our protocol we abstract away the implementation of digital cash used, and simply describe a digital coin as consisting of a triple

[1] More precisely, our multicast protocols, which employ timeouts in their methods for failure detection, satisfy the stated specifications despite t failures in a group of size $3t+1$ provided that messages from correct members induce timeouts in other correct members sufficiently infrequently. See [21] for details.

$(v_\$, \sigma_{\mathsf{bank}}(v_\$), w_\$)$, where $v_\$$ is a description of the coin, $\sigma_{\mathsf{bank}}(v_\$)$ is the signature of the bank on that description, and $w_\$$ is some auxiliary information that must accompany the coin when it is used in a purchase. The description $v_\$$ would typically include the value of the coin, and an embedding of the customer's identity as described above. The auxiliary information $w_\$$ would typically be a "hint," any two of which enable the extraction of the embedded identity, and would include certain freshness information so that a vendor can detect the replay of a coin. Our auction protocol requires that the procedure for a vendor to determine the validity of $v_\$$ and $w_\$$ be a deterministic function of these values that it can compute locally.

Cash schemes can differ in the particular signature scheme used by the bank to sign coins (e.g., RSA [23]). This will have an impact on the number of faults that can be tolerated in our auction protocol, as will be discussed in Section 4.

4 Verifiable signature sharing

In addition to the primitives reviewed in Section 3, our auction protocol employs a new cryptographic primitive for protecting digital signatures, called *verifiable signature sharing* (VΣS). VΣS enables the holder of a digitally signed message, who need not be the original signer, to share the signature among a group of processes so that the correct members can later reconstruct it. At the end of the sharing phase, each member can verify whether a valid signature for the message can be reconstructed, even if the original signature holder and/or a constant fraction of the members are malicious. In addition, malicious members gain no information prior to reconstruction about the signature held by a (correct) sharer. In a separate paper [14], we present and prove the correctness of efficient VΣS schemes for many signature schemes, including RSA [23], Rabin [20], ElGamal [10], Schnorr [24], and DSA [9]. In the present work, our purpose is to present VΣS only to the extent necessary to describe our auction service.

Our auction protocol does not rely on any particular VΣS scheme, and so for generality, here we describe VΣS in an abstract form. If B holds a signature $\sigma(m)$ of a message m (i.e., $\sigma(m) = \sigma_P(m)$ for some P), then B begins the VΣS protocol by generating two types of values from $\sigma(m)$: a public value VΣS-pub$(\sigma(m))$ and, for each process S_i in the group $\mathcal{G}$ among which the signature is to be shared, a private value VΣS-priv$_i(\sigma(m))$. B then atomically multicasts[2] VΣS-pub$(\sigma(m))$ to the group members and communicates VΣS-priv$_i(\sigma(m))$ to S_i privately, say, encrypted under the public key K_i for S_i:

$$B \xrightarrow{A} \mathcal{G}: \quad m,\ \text{V}\Sigma\text{S-pub}(\sigma(m)),\ \{\langle \text{V}\Sigma\text{S-priv}_j(\sigma(m))\rangle_{K_j}\}_{S_j \in \mathcal{G}}$$

Upon receipt of such an atomic multicast, S_i performs a local computation to determine whether the i-th private value (which it decrypts with K_i^{-1}) is consistent with the public value. S_i reliably multicasts the status of this computation, denoted VΣS-stat$_i$, to the group:

$$S_i \xrightarrow{R} \mathcal{G}: \quad \text{V}\Sigma\text{S-stat}_i$$

Finally, once S_i has received a reliable multicast from S_j (or detected S_j faulty) for each $S_j \in \mathcal{G}$, it performs a local computation that allows it to either *accept* or *reject* the attempt to share $\sigma(m)$. This local computation is a deterministic function of the reliably and atomically multicast values only, and so either all correct group members accept or all correct members reject. If they accept, then this guarantees that $\sigma(m)$ can be reconstructed with the information they collectively possess. If $\sigma(m)$ was shared correctly, then the correct members will accept, but faulty members gain no information about $\sigma(m)$. If at some point the correct members choose to reconstruct $\sigma(m)$, they can do so by each member S_i forwarding its private value VΣS-priv$_i(\sigma(m))$ (and possibly some other auxiliary information) to the reconstructing party, which can then easily reconstruct the signature.

As an example of a VΣS scheme, here we outline the VΣS scheme for RSA described in [14]. This scheme can tolerate up to t malicious group members in a group of size n (in addition to a malicious signature holder) whenever $n \geq 5t + 1$. In this scheme, $\sigma(m) = (h(m))^d \bmod N$ where h is a message digest function (e.g., MD5 [22]) and where N and d are the RSA modulus and private exponent for the signer; the public (verifying) exponent is $e = 3$. To share $\sigma(m)$ to a group $\mathcal{G} = \{S_1, \ldots, S_n\}$ with a tolerance of t member failures, B chooses a random degree t polynomial $f \in Z_N[x]$ such that $f(0) = \sigma(m)$, and sets VΣS-pub$(\sigma(m)) = \{f(j)^3 \bmod N\}_{S_j \in \mathcal{G}}$ and VΣS-priv$_j(\sigma(m)) = f(j) \bmod N$. That is, B executes

$$B \xrightarrow{A} \mathcal{G}: \quad m,\ \{f(j)^3 \bmod N\}_{S_j \in \mathcal{G}},\ \{(f(j) \bmod N)_{K_j}\}_{S_j \in \mathcal{G}}$$

[2] A weaker multicast can be used (see [14]), but we use atomic multicast here for consistency with the protocol of Section 5 and due to nuances of our multicast specifications in Section 3.1.

Continuing, now suppose that S_i receives an atomic multicast of the form

$$m, \{u_j\}_{S_j \in \mathcal{G}}, \{y_j\}_{S_j \in \mathcal{G}}$$

for some values of m, $\{u_j\}_{S_j \in \mathcal{G}}$, and $\{y_j\}_{S_j \in \mathcal{G}}$. S_i computes VΣS-stat$_i$ as

$$\text{V}\Sigma\text{S-stat}_i = \begin{cases} \langle \text{allow}, r_i \rangle & \text{if } u_i = (\langle y_i \rangle^{K_i^{-1}})^3 \bmod N \\ \langle \text{complain}, r_i \rangle & \text{otherwise} \end{cases}$$

where $r_i = \langle y_i \rangle^{K_i^{-1}} + p(i) \bmod N$ and $p(i)$ is S_i's private share of a secret, random degree t polynomial $p \in Z_N[x]$; $p(i)$ is given to S_i at initialization. S_i reliably multicasts VΣS-stat$_i$ and collects status values from the other servers. Finally, S_i accepts if the values $\{u_j\}_{S_j \in \mathcal{G}}$ lie on a polynomial g of degree at most $3t$, $g(0) = h(m)$, the values $\{r_j\}_{S_j \in \mathcal{G}}$ lie on a polynomial g' of degree t with at most t errors (any absent r_j, due to S_j failing, is counted as an error), and at most t processes complained or contributed an error to g'.

All VΣS schemes in [14] provide protection against a constant fraction of faulty members, but the constant is different for different schemes. As mentioned above, n must be greater than $5t + 1$ for RSA, but it must be greater than only $3t + 1$ for signature schemes based on discrete logarithms (i.e., ElGamal, Schnorr, and DSA). Since in either case the value for n that is necessary to tolerate t member failures is at least that required for the multicast protocols of Section 3.1, the fault tolerance of our auction scheme will be equal to the fault tolerance of the VΣS scheme used in it. The choice of VΣS scheme depends on the signature scheme used by the bank to sign its digital cash.

5 The auction protocol

Our auction service is constructed using n auction servers. There is a parameter t that defines the fault tolerance of the service, i.e., the maximum number of servers that can fail without affecting the correctness of the service. As described in Section 4, our protocol allows t to be as large as a constant fraction of n, which depends on the VΣS scheme, and thus the digital cash scheme, used.

Intuitively, our auction protocol works as follows. A bidder submits a bid of a certain value to the service by sharing the pieces of a digital coin $\langle v_\$, \sigma_{\mathsf{bank}}(v_\$), w_\$ \rangle$ with that value among the auction servers. The description $v_\$$ and auxiliary information $w_\$$ are shared with a standard (t, n)-threshold secret sharing scheme (see Section 3.2), while the signature $\sigma_{\mathsf{bank}}(v_\$)$ is shared with a VΣS scheme (see Section 4). Once the bidding period has closed, the servers reconstruct $v_\$$ and $w_\$$ for each bid received during the bidding period, and then perform the VΣS protocol to determine acceptance or rejection for each bid (i.e., to determine if they collectively possess $\sigma_{\mathsf{bank}}(v_\$)$). The servers then choose the winning bid from the acceptable bids and declare the winner. Finally, subject to auction house controls, the bank's signature on the coin in the winner's bid can be reconstructed via the VΣS scheme, and the coin can be deposited. The secrecy of each bid is ensured until after bidding is closed because correct servers do not cooperate in the reconstruction of $v_\$$ and $w_\$$ until after bidding is closed. Moreover, since $\sigma_{\mathsf{bank}}(v_\$)$ is never reconstructed for a losing bid, the coins in losing bids cannot be spent by faulty servers.

In Section 5.1, we describe this protocol in more depth. In Section 5.2 we discuss alternative designs that we considered and compare them to our protocol.

5.1 The protocol detailed

In this section we more carefully describe the auction protocol. For simplicity, our description assumes only a single auction; executing concurrent auctions at the same auction service requires modifications to the protocol and is discussed in Section 8.2. The n auction servers, denoted by $S_1, \ldots, S_n$, are organized as a process group $\mathcal{G}$ to which processes can multicast messages (unreliably, reliably, or atomically). Associated with each server S_i is a public key K_i for use in a deterministic public key cryptosystem (e.g., RSA [23]). Each K_i is assumed to be available to all servers and bidders; the corresponding private key K_i^{-1} known only to S_i. In addition, we assume that a global identifier *aid* for the auction is known by all servers and bidders. In the description below, $||$ denotes concatenation. We remind the reader that only multicasts from servers (i.e., members of $\mathcal{G}$) are authenticated, and that $sh_i(s)$ denotes the i-th share of s produced via Shamir's (t, n)-threshold secret sharing scheme.

Submitting a bid Suppose a bidder wishes to submit a bid to the auction. Without loss of generality, we assume that the bidder possesses a digital coin $\langle v_\$, \sigma_{\mathsf{bank}}(v_\$), w_\$ \rangle$ in the amount of the desired bid. The freshness information included in $w_\$$ (see Section 3.3) is *aid*. The bidder B submits the bid using a single atomic multicast as follows:

$$\text{(M1)} \quad B \xrightarrow{A} \mathcal{G}: \quad \{\langle sh_j(B||v_\$||w_\$) \rangle_{K_j}\}_{S_j \in \mathcal{G}}, \text{V}\Sigma\text{S-pub}(\sigma_{\mathsf{bank}}(v_\$)), \{\langle \text{V}\Sigma\text{S-priv}_j(\sigma_{\mathsf{bank}}(v_\$)) \rangle_{K_j}\}_{S_j \in \mathcal{G}}$$

Closing the bidding period When server S_i decides that bidding should be closed, it executes:

(M2) $S_i \xrightarrow{A} \mathcal{G}$: close

When S_i has received (by atomic multicast) close messages from $t+1$ different servers, it considers bidding closed and ignores any bids subsequently received. Note that by the properties of atomic multicast, all correct servers will agree on the set of bids received prior to closing.

Opening the bids Suppose that the l-th bid received (by atomic multicast) at S_i is of the form

$$\{c_{j,l}\}_{S_j \in \mathcal{G}},\ \text{pub}_l,\ \{\text{priv}_{j,l}\}_{S_j \in \mathcal{G}} \qquad (*)$$

for some values of $\{c_{j,l}\}_{S_j \in \mathcal{G}}$, pub_l, and $\{\text{priv}_{j,l}\}_{S_j \in \mathcal{G}}$. Also suppose that a total of L bids were received. These bids are opened in three steps:

1. S_i computes $s_{i,l} = \langle c_{i,l} \rangle^{K_i^{-1}}$ for each l, $1 \le l \le L$, and executes:

   (M3) $S_i \rightarrow \mathcal{G}$: $\{s_{i,l}\}_{1 \le l \le L}$

2. When S_i receives a message of the form $\{s_{j,l}\}_{1 \le l \le L}$ from S_j, it verifies that $\langle s_{j,l} \rangle_{K_j} = c_{j,l}$ for each l, $1 \le l \le L$. If there is an l for which this does not hold, then S_i discards this message from S_j and ignores it. Note that if this occurs, then S_j is faulty.

3. Once S_i has received $t+1$ messages from $t+1$ different servers $S_{j_1}, \ldots, S_{j_{t+1}}$ that pass these verifications, then for each l, $1 \le l \le L$, S_i finds the degree t polynomial f_l determined by the $t+1$ values $s_{j_1,l} \ldots s_{j_{t+1},l}$. S_i then verifies that $\langle f_l(j) \rangle_{K_j} = c_{j,l}$ for each j such that $1 \le j \le n$ and $j \ne j_1, \ldots, j_{t+1}$. If for any such j this does not hold, then S_i discards the l-th bid. S_i also discards the l-th bid if $f_l(0)$ is not of the form $B_l || v_{\$,l} || w_{\$,l}$ for values B_l, $v_{\$,l}$, and $w_{\$,l}$ of a proper syntactic form. Note that if the l-th bid is discarded, then it must have been submitted by a faulty bidder. Let the bids that remain be renumbered $1 \le l \le L'$.

Checking the validity of bids S_i checks the validity of the remaining bids as follows.

1. For each l, $1 \le l \le L'$, S_i first performs the validity checks on $v_{\$,l}$ and $w_{\$,l}$ that are dictated by the electronic money scheme in use, discarding any bid that is found to be invalid or a replay. By the properties of the off-line cash scheme and the choice of freshness information embedded in $w_{\$}$, these tests involve only local deterministic computations. Let the remaining bids be renumbered $1 \le l \le L''$.

2. S_i computes $\text{V}\Sigma\text{S-stat}_{i,l}$ (from $\langle \text{priv}_{i,l} \rangle^{K_i^{-1}}$ and pub_l; see $(*)$) for each l, $1 \le l \le L''$, according to the VΣS scheme, and executes

   (M4) $S_i \xrightarrow{R} \mathcal{G}$: $\{\text{V}\Sigma\text{S-stat}_{i,l}\}_{1 \le l \le L''}$

   S_i collects reliable multicasts from the other servers and determines acceptance or rejection for each remaining bid according to the VΣS scheme (see Section 4). All rejected bids are discarded.

Declaring the winner Server S_i chooses the winning bid from among the remaining bids. Once the winning bidder B is determined, S_i executes

(M5) $S_i \rightsquigarrow B$: $aid, B, \sigma_{S_i}(aid || B)$

where $\rightsquigarrow$ denotes a point-to-point send over a (not necessarily authenticated) communication channel. This message conveys that S_i declares B the winner of auction *aid*. Once B receives such messages from $t+1$ different servers, it can claim the item bid upon by presenting the $t+1$ signed declarations to the appropriate authority.

At this point, correct servers can erase any information they hold for losing bids. By the properties of the VΣS scheme, the correct servers possess enough information to reconstruct the bank's signature for the coin used in the winning bid. However, the servers should not reconstruct this signature among themselves, lest a faulty server reconstruct and deposit the coin in its own account before the correct servers can deposit the coin in the auction's account. Moreover, as discussed in Section 8.1, enabling faulty servers to reconstruct the coin's signature might allow them to "frame" a bidder for reusing the coin, if the bidder does not take recommended precautions. To perform reconstruction in a way that avoids these problems, each server could forward its private VΣS value for that coin's signature to the bank, so that the bank can perform the reconstruction itself. This would result in minimal overhead at the bank (see [14]), but it requires the bank to provide an interface to support VΣS reconstruction. Since reconstruction can occur at any time after the auction (e.g., at the end of the day), an alternative

solution would be for reconstruction to be performed later by a trusted financial officer of the auction house.

5.2 Alternative designs

In the design of our auction protocol, we considered numerous alternatives to that presented in Section 5.1, and it is instructive to discuss several of them.

Eliminating VΣS It is possible to eliminate the use of a VΣS scheme by having the bidder share $\sigma_{\mathsf{bank}}(v_\$)$ among the servers with a standard threshold secret sharing scheme or a *verifiable* secret sharing scheme [6]. In this case, the auction servers would have no way of verifying that they hold shares of a proper signature, except by reconstructing it. Reconstructing it, however, would leave the coin vulnerable to theft, by a faulty server depositing the coin in its own account before the correct servers could deposit it in the auction's account.

Even if it were deemed acceptable to simply minimize the number of coins exposed to theft, doing so would require that the servers locate the highest bid containing a valid coin by reconstructing the signatures in the sorted bids one (or a few) at a time, until that bid is found. In this approach, the message complexity of finding the highest valid bid can be proportional to the number of invalid bids submitted. Therefore, it is susceptible to an explosion in communication costs if malicious bidders submit a large number of invalid bids. Moreover, this attack would be very difficult to prevent or punish, especially since bids are not authenticated (to withhold the identities of the bidders until after bidding is closed) and may even be anonymous (see Section 8.1).

These problems are avoided with the use of a VΣS scheme. In our protocol, no coins are exposed to theft by faulty servers, and the validity of *all* bids can be checked with a total of n reliable multicasts. Moreover, the use of VΣS makes it possible to extend our auction protocol to perform auctions in which the amount the winner pays is a function of other valid bids (e.g., second-price sealed-bid auctions [18]). Implementing such auctions with only the mechanism described above would force servers to reconstruct the coins in those other bids, thus exposing them to theft.

On-line digital cash It is conceivable that our protocol could be modified to accommodate the use of on-line digital cash. With an on-line cash scheme, checking the validity of a bid would involve the bank, typically to determine whether the coin in a bid was previously spent. Unfortunately, most obvious approaches to performing this interaction with the bank either expose coins to theft by a faulty server or result in a message complexity that depends on the number of invalid bids. While it is possible to overcome these difficulties, doing so seems to require substantial changes to the interface provided by the bank in a typical on-line cash scheme (e.g., [4, 19]).

Threshold cryptography In our auction protocol, the technique used to keep the bids secret prior to the close of bidding is to share the value of the bid among the auction servers using a threshold secret sharing scheme. Alternatively, a threshold public key cryptosystem [7] could be used to encrypt bids under the public key of the auction house, so that they could be decrypted only with the cooperation of a threshold number of servers. Correct servers could prevent the premature disclosure of bids by cooperating in decryptions only after bidding had closed. The primary drawback of this approach is that with all threshold cryptosystems of which we are aware, a large modular exponentiation would be required per server *per bid.* Since modular exponentiations are computationally intensive, this would expose the service to substantial computational overheads induced by malicious bidders submitting large numbers of bids. Such an attack would be much less effective against our protocol, because to open bids, the main cost per server per bid is a polynomial interpolation, which is relatively inexpensive (a small number of linear combinations using precomputed coefficients; see Section 7).

A threshold signature scheme [7], in which the cooperation of a threshold number of servers is required to sign a message with the auction house's private key, could be useful when declaring a winner. Instead of sending separate signed messages to the winning bidder in step (M5), the servers could construct a single ticket bearing the auction house's signature and send this to the winner. This would decrease the size of the ticket that the winner must present to claim the auctioned item, but, with existing threshold signature schemes, would also increase the computational load on the servers to construct this ticket.

Mental games "Mental games" [15] are known cryptographic techniques for securely performing a wide variety of tasks, including secure auctions as a special case. Mental games could be used to construct an auction service that provides stronger properties than ours—e.g., that the values of bids are *never* disclosed, even after bidding closes—but a service built using these techniques would perform much worse

than ours. Our protocol sacrifices the above property in the interest of efficiency, although our protocol can be modified to allow bidder's *identities* to remain secret even after bidding closes; see Section 8.1.

6 Security

In this section, we discuss how the protocol of Section 5.1 achieves the security properties stated in Section 2. Our arguments are informal, and are not intended to constitute a proper proof of security. Such a proof would be quite involved, and would require more detailed definitions of security for auctions and the properties of VΣS and multicast.

Validity

1. *The bidding period eventually closes, but only after a correct auction server decides that it should be closed.*

   The bidding period eventually closes because all correct servers (and thus at least $t + 1$ correct servers) atomically multicast close messages. Moreover, since the bidding period closes at each server after it has received (by atomic multicast) close messages from $t + 1$ servers, the bidding period closes at a server only after it has received a close message from a correct server.

2. *There is at most one winning bid per auction, dictated by the (deterministic) publicly-known rule applied to the well-formed bids received before the end of the bidding period.*

   All correct servers agree on which bids were received before the close of the bidding period, due to the properties of atomic multicast. If the servers received correct shares of $B_l||v_{\$,l}||w_{\$,l}$ in the l-th bid, then all correct servers will reconstruct $B_l||v_{\$,l}||w_{\$,l}$, since any bad shares provided by faulty servers will be discarded in step 2 of "opening the bids". In addition, if the sharing was not performed correctly, then the bid will be discarded by all correct servers, as all will detect the discrepancies between the interpolated polynomial f_l and the values $\{c_{j,l}\}_{S_j \in \mathcal{G}}$ (step 3 of "opening the bids"). Thus all correct servers agree on the same set of L' bids after opening the bids. By the properties of the digital cash and VΣS schemes, all correct servers will agree on the subset of those bids that pass the validity checks. All correct servers will select the same winning bid from these acceptable bids, by following the public rule for determining the winner. Finally, all correct servers will sign a message announcing this winning bid, enabling the winner to claim the item bid upon.

3. *The auction service collects payment from the winning bidder equal to the amount of the winning bid.*

   By the properties of the VΣS scheme, the correct servers end the protocol in possession of shares sufficient to reconstruct the bank's signature on the coin contained in the winning bid. This signature can be reconstructed via the VΣS reconstruction protocol, according to auction house policy.

4. *Correct losing bidders forfeit no money.*

   The money from a losing bid is worthless without the bank's signature. By the properties of VΣS, no information about this signature is leaked to a coalition of faulty servers, and so the faulty servers are unable to deposit the money. Thus, the money is effectively transferred back to the bidder, who can reuse the money as it chooses.

5. *Only the winning bidder can collect the item bid upon.*

   Only the winning bidder obtains $t+1$ signed declarations (from $t+1$ different auction servers) stating that it won the auction. Thus, only the winning bidder can collect the item bid upon, supposing that possession of $t + 1$ such declarations is necessary to do so.

Secrecy

1. *The identity of a correct bidder and the amount of its bid are not revealed to any party until after the bidding period is closed.*

   More precisely, the identity of the bidder and the amount of the bid are not revealed until after the bidding period is closed *at some correct server.* This prevents bids being submitted based on the previously disclosed contents of other bids, because by the properties of atomic multicast, once bidding is closed at any correct server, the set of bids that will be considered by any correct server is fixed.

   Showing the stated property is not straightforward, since it depends on additional properties of VΣS and digital cash schemes. Intuitively, however, a coalition of faulty servers cannot reconstruct the value $B||v_{\$}||w_{\$}$ shared in a bid from

only their shares of this value, by the properties of threshold secret sharing schemes. Moreover, in all of the VΣS implementations proposed in [14], the public and private VΣS information available to the coalition would yield at most the message digest of $v_\$$. In a typical digital cash scheme, $v_\$$ includes a large, unpredictable component, such as a string with the coinholder's identity embedded in it. Thus, the message digest of $v_\$$ reveals no useful information about the amount of the bid.

7 Performance

We have implemented a prototype of our auction service using the protocol of Section 5.1, in an effort to understand the factors that limit its performance. Our implementation uses the multicast protocols of Rampart [21], and employs Cryptolib [16] for basic cryptographic operations. Our implementation includes many optimizations to the protocol described in Section 5.1. For example, to avoid sharing the entire value $B||v_\$||w_\$$ when submitting a bid, we share a (much smaller) key to a symmetric cipher and include in the bid the encryption of $B||v_\$||w_\$$ under that key. In addition, since each server must receive only $t+1$ shares of this key to recover $B||v_\$||w_\$$, only $2t+1$ servers multicast shares and, in fact, only $2t+1$ shares are distributed by the bidder for each bid. Similarly, only $2t+1$ servers multicast close messages, as this suffices to ensure that each correct server receives close messages from $t+1$ servers.

Latency numbers in milliseconds (ms) for the stages of the auction protocol in the case of no failures are shown in Table 1 and Figure 1. These numbers were taken on a network of moderately loaded SPARCstation 10s. These tests used RSA for public key encryption and signatures, with 512-bit moduli and public exponents equal to three; thus, the RSA-based VΣS scheme of [14] (see Section 4) was also used. There were six auction servers, which is the minimum number of servers required to tolerate the failure of one auction server with this VΣS scheme (i.e., $t=1$).

In order to isolate the costs of our auction protocol, the numbers in Table 1 and Figure 1 do not reflect operations specific to the form of digital cash used. In particular, the latencies labeled "submitting a bid" in Table 1 and "checking the validity of bids" in Figure 1 do *not* include the costs of creating $w_\$$ and checking the validity of $v_\$$ and $w_\$$, respectively. For the purposes of interpreting these test results, $v_\$$ and $w_\$$ can together be viewed as a single opaque 256-byte string, a size comparable to that in modern off-line cash schemes (e.g., [3] using a 512-bit modulus).

Operation	Latency (ms)
Submitting a bid	261
Closing the bidding period	230
Declaring the winner	62

Table 1: Operations with constant latency

Table 1 shows operations whose latencies are relatively constant as a function of the number of bids submitted to the auction. The latency labeled "submitting a bid" includes the latencies of creating a bid, atomically multicasting it to the server group, and each server, in parallel, decrypting the two portions of the message private to it. "Closing the bidding period" is the latency of parallel atomic multicasts (close messages) from $2t+1$ servers. "Declaring the winner" includes the latencies of each server, in parallel, signing the message declaring the winner and sending it, and the winner receiving and verifying the signatures on $t+1$ such messages.

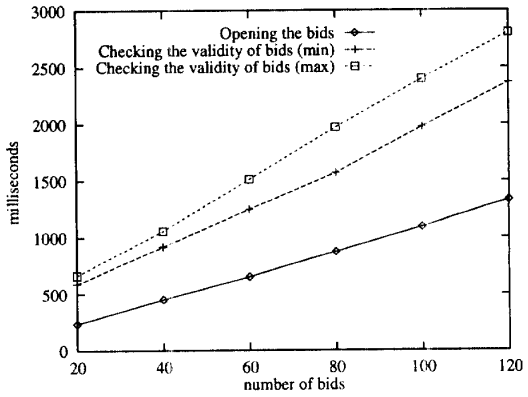

Figure 1: Latencies that grow with number of bids

Figure 1 shows operations whose latencies increase as a function of the number of bids submitted. "Opening the bids" includes the latency of $2t+1$ servers, in parallel, unreliably multicasting messages containing their previously decrypted shares for the value $B||v_\$||w_\$$ for each bid, and all servers receiving $t+1$ such messages and reconstructing these values as described in Section 5.1. "Checking the validity of bids (max)" includes the latency of each server, in parallel, computing its VΣS-stat values for all bids, reliably multicasting these values, and completing the verifica-

tion for each bid, until each is either accepted or rejected. "Checking the validity of bids (min)" includes the latency of this procedure only until the highest bid, whose verification is completed first, is found to be acceptable. Thus, these curves form a minimum and maximum latency for finding the winning bid in our implementation, once the bids are opened. Since our implementation is intended to minimize the "max" curve (versus the "min" curve), it should be possible to find implementations of our protocol that provide a better "min" curve than that shown in Figure 1.

8 Extensions

8.1 Anonymity

As discussed in Section 3.3, a goal of most approaches to electronic money is to provide anonymous spending to customers, i.e., to prevent a vendor or bank from associating purchases to individuals. In this section, we discuss the ability of a bidder to retain that anonymity in the auction protocol.

An immediate requirement to achieving bidder anonymity is to remove the identity of the bidder from the protocol of Section 5.1. A simple approach to achieve this is for each bidder, prior to submitting a bid, to generate a large random number r and use $h(r)$ as a pseudonym for that bid, where h is a message digest function (e.g., MD5). That is, a bid would be submitted as

$$\text{(M1')}\quad B \xrightarrow{A} \mathcal{G}:\quad \{\langle sh_j(h(r)||v_\$||w_\$)\rangle_{K_j}\}_{S_j\in\mathcal{G}},\ \text{V}\Sigma\text{S-pub}(\sigma_{\mathsf{bank}}(v_\$)),\ \{\langle \text{V}\Sigma\text{S-priv}_j(\sigma_{\mathsf{bank}}(v_\$))\rangle_{K_j}\}_{S_j\in\mathcal{G}}$$

The auction would then proceed as before, except that the winner would be announced as follows:

$$\text{(M5')}\quad S_i \text{ broadcasts:}\quad aid, h(r), \sigma_{S_i}(aid, h(r))$$

Note that S_i, not knowing the identity or location of the bidder that submit the bid with pseudonym $h(r)$, must simply broadcast the declaration of the winner. Alternatively, S_i could place this signed declaration in a location from which it could be later retrieved by the winning bidder. Once the winning bidder has obtained $t+1$ such signed declarations, it can claim the item bid upon by presenting these declarations and the number r to the appropriate authority; since h is a message digest function, the winning bidder is the only party that could produce r.

While at first this may seem to ensure the bidder's anonymity, other steps may be required due to the properties of off-line digital cash. As discussed in Section 3.3, off-line cash schemes require that the customer's (in this case, the bidder's) identity be embedded within the value $v_\$$ in a way that reveals this identity to the bank if the same coin is spent multiple times. Thus, with proposed off-line cash schemes, if a bidder were to submit the same coin to two auctions (e.g., submit the coin to one, lose the auction, and submit the coin to another), then the identity of the bidder could be inferred by a coalition of one faulty auction server from each auction. Perhaps even worse, if $\sigma_{\mathsf{bank}}(v_\$)$ is ever leaked to the coalition of faulty servers (e.g., due to a weakness in the procedures by which the coin is reconstructed and deposited after it wins the second auction), then they could deposit both uses of the coin, thereby revealing the bidder's identity to the bank and "framing" the bidder for reusing the coin. It is possible to modify proposed off-line cash schemes so that the identity information embedded in $v_\$$ is encrypted with a key known only to the bank and the bidder. Then, the bank's cooperation would be required to reveal the identity of the bidder. However, this approach still enables the coalition of auction servers to link the same coin, and thus the same (unknown) bidder, to both auctions, and does not prevent the "framing" attack described above.

There are steps that a bidder should take to guard against these attacks. Specifically, the bidder should use a coin in at most one bid. If that bid is unsuccessful, the bidder should deposit the coin in the bank and withdraw a new one. In this case, multiple bids cannot be linked to the same bidder or used to frame the bidder for reuse, and the identity of the bidder can be revealed only by a coalition involving the bank and a faulty auction server. However, it is not clear how a bidder can conceal its identity against such a coalition with current off-line schemes.

8.2 Concurrent auctions

The protocol of Section 5.1 does not suffice to handle multiple auctions with overlapping bidding periods at the same auction service, because a bid conveys the auction for which it is intended only after $B||v_\$||w_\$$ is reconstructed (and Secrecy has possibly been violated). Thus, to adapt the protocol to accommodate concurrent auctions, the format of a bid must be modified to enable all correct servers to determine the auction for which a bid is intended. A first attempt might be for the bidder to include *aid*, in plaintext, in the body of message M1. However, since M1 is not authenticated by the servers, an attacker could modify *aid* in transit, replacing it with the identifier of an auction

whose bidding period closes earlier. This would be detected by the auction servers, but only after $B||v_\$||w_\$$ was reconstructed and Secrecy had been violated.

To rectify this problem, it is necessary to bind *aid* to the shares of $B||v_\$||w_\$$, so that servers can detect any modification to *aid* before reconstructing $B||v_\$||w_\$$. More precisely, we propose the following bid format:

$$\text{(M1}'') \quad B \xrightarrow{A} \mathcal{G}: \quad aid, \{\langle sh_j(B||v_\$||w_\$)||aid\rangle_{K_j}\}_{S_j \in \mathcal{G}}, \; \text{V}\Sigma\text{S-pub}(\sigma_{\text{bank}}(v_\$)), \; \{\langle \text{V}\Sigma\text{S-priv}_j(\sigma_{\text{bank}}(v_\$))\rangle_{K_j}\}_{S_j \in \mathcal{G}}$$

When server S_i decrypts $\langle sh_i(B||v_\$||w_\$)||aid\rangle_{K_i}$, it verifies that the value of *aid* appended to $sh_i(B||v_\$||w_\$)$ matches the value of *aid* at the front of the bid. If these values do not match, then S_i does not reveal $sh_i(B||v_\$||w_\$)$ during the "opening the bids" phase. However, it cannot completely abandon the protocol for opening this bid, because other correct servers may have found a matching pair of *aid* values. Therefore, we must modify the "opening the bids" step of the protocol as described below, to ensure that all correct servers consistently accept or reject this bid.

Opening the bids Suppose that the l-th bid for auction *aid* received (by atomic multicast) at S_i is of the form

$$aid, \; \{c_{j,l}\}_{S_j \in \mathcal{G}}, \; \text{pub}_l, \; \{\text{priv}_{j,l}\}_{S_j \in \mathcal{G}}$$

for some values of $\{c_{j,l}\}_{S_j \in \mathcal{G}}$, pub_l, and $\{\text{priv}_{j,l}\}_{S_j \in \mathcal{G}}$. Also suppose that a total of L bids were received for auction *aid*. These bids are opened in three steps:

1. For each l, $1 \le l \le L$, server S_i computes

$$s_{i,l} = \begin{cases} q & \text{if } \langle c_{i,l}\rangle^{K_i^{-1}} = q||aid \\ \perp & \text{otherwise} \end{cases}$$

   S_i then executes:

$$\text{(M3}'') \quad S_i \rightarrow \mathcal{G}: \; aid, \{s_{i,l}\}_{1 \le l \le L}$$

2. When S_i receives a message of the form

$$aid, \; \{s_{j,l}\}_{1 \le l \le L} \qquad (\dagger)$$

   from a server S_j, it verifies for each l, $1 \le l \le L$, that if $s_{j,l} \neq \perp$, then $\langle s_{j,l}||aid\rangle_{K_j} = c_{j,l}$. If there is an l for which this does not hold, then S_i discards and ignores this message from S_j. Note that if this occurs, then S_j must be faulty.

3. S_i completes the opening of the l-th bid, $1 \le l \le L$, as follows. If in the first $2t+1$ messages of the form (†) that S_i receives (from different servers, and that pass the verifications of step 2), there are $t+1$ messages, say from $S_{j_1}, \ldots, S_{j_{t+1}}$, such that $s_{j_k,l} \neq \perp$ for all k, $1 \le k \le t+1$, then S_i finds the degree t polynomial f_l determined by $s_{j_1,l} \ldots s_{j_{t+1},l}$. S_i then verifies that $\langle f_l(j)||aid\rangle_{K_j} = c_{j,l}$ for all j such that $1 \le j \le n$ and $j \neq j_1, \ldots, j_{t+1}$. If for any such j this does not hold, then S_i discards the l-th bid. S_i also discards the l-th bid if $f_l(0)$ is not of the form $B_l||v_{\$,l}||w_{\$,l}$ for some B_l, $v_{\$,l}$, and $w_{\$,l}$ of a proper syntactic form, or if in the first $2t+1$ messages of the form (†) that S_i receives, there are $t+1$ messages, say from $S_{j_1}, \ldots, S_{j_{t+1}}$, such that $s_{j_k,l} = \perp$ for all k, $1 \le k \le t+1$. Note that if the l-th bid is discarded, then it was submitted by a faulty bidder.

The rest of the auction protocol remains as it is described in Section 5.1, except that messages M2 and M4 must be prepended with *aid* to mark the auction to which they pertain.

All correct servers agree on the remaining set of bids after the above "opening the bids" step, due to the following two observations. First, if any correct server S_i sends $s_{i,l} = \perp$, then each correct server S_j will discard the l-th bid, because either S_j will receive too many $\perp$ values to determine an f_l or otherwise will notice that $\langle f_l(i)||aid\rangle_{K_i} \neq c_{i,l}$. Second, if $s_{i,l} \neq \perp$ for each correct S_i, then each correct server will determine an f_l and will consistently accept or reject the bid as in the protocol of Section 5.1. Also note that any attempt by an attacker to redirect a bid to an earlier auction will result in each correct server contributing $\perp$ to the bid opening, thus causing the bid to be rejected before the amount or source of the bid are revealed.

9 Conclusion

We have presented the design and implementation of a practical distributed auction service that can tolerate the malicious behavior of a constant fraction of its servers and any number of bidders. Our design is based on several cryptographic primitives, both old (multicast, secret sharing, digital cash) and new (verifiable signature sharing). Our implementation of this service suggests that this approach performs sufficiently well to be useful in a wide range of settings.

As described in Section 1, this work is part of a larger effort to understand how to implement common financial vehicles in distributed systems. We are

continuing in this effort, and plan to extend the techniques developed in this work to address more general types of auctions and other financial vehicles. We hope to report on this work in future papers.

Acknowledgements

The first author would like to thank Stuart Haber for early discussions on cryptographic auctions. We also thank the anonymous referees for presentational suggestions.

References

[1] G. R. Blakely. Safeguarding cryptographic keys. In *Proceedings of the AFIPS National Computer Conference*, pages 313–317, 1979.

[2] J. Boly et al. The ESPRIT project CAFE—high security digital payment system. In D. Gollmann, editor, *Computer Security—ESORICS 94* (Lecture Notes in Computer Science 875). Springer-Verlag, 1994.

[3] S. Brands. Untraceable off-line cash in wallets with observers. In D. R. Stinson, editor, *Advances in Cryptology—CRYPTO '93* (Lecture Notes in Computer Science 773), pages 302–318. Springer-Verlag, 1994.

[4] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28:1030–1044, 1985.

[5] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In S. Goldwasser, editor, *Advances in Cryptology—CRYPTO '88 Proceedings* (Lecture Notes in Computer Science 403), pages 319–327. Springer-Verlag, 1989.

[6] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 383–395, October 1985.

[7] Y. Desmedt. Threshold cryptography. *European Transactions on Telecommunications and Related Technologies*, 5(4):449–457, July 1994.

[8] W. Diffie. The impact of a secret cryptographic standard on encryption, privacy, law enforcement and technology. Hearings before the Subcommittee on Telecommunications and Finance of the Committee on Energy and Commerce, House of Representatives, One Hundred Third Congress, First Session, April 29 and June 9, 1993, Serial No. 103-53, pp. 111-116.

[9] NIST FIPS PUB 181, Digital Signature Standard. U.S. Department of Commerce/National Institute of Standards and Technology.

[10] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, July 1985.

[11] S. Even, O. Goldreich, and Y. Yacobi. Electronic wallet. In *Proceedings of CRYPTO '83*. Plenum Press, 1984.

[12] FCC takes licenses, denies more time to 13 bidders. AP-Dow Jones News, August 10, 1994.

[13] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 427–437, October 1987.

[14] M. K. Franklin and M. K. Reiter. Verifiable signature sharing. In *Advances in Cryptology—EUROCRYPT '95*. Springer-Verlag, 1995. To appear.

[15] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th ACM Symposium on the Theory of Computing*, pages 218–229, May 1987.

[16] J. B. Lacy, D. P. Mitchell, and W. M. Schell. CryptoLib: Cryptography in software. In *Proceedings of the 4th USENIX Security Workshop*, pages 1–17, October 1993.

[17] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[18] R. McAfee and J. McMillan. Auctions and bidding. *Journal of Economic Literature*, 25:699–738, June 1987.

[19] G. Medvinsky and B. C. Neuman. NetCash: A design for practical electronic currency on the Internet. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 102–106, November 1993.

[20] M. O. Rabin. Digitalized signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Laboratory for Computer Science, Massachusetts Institute of Technology, January 1979.

[21] M. K. Reiter. Secure agreement protocols: Reliable and atomic group multicast in Rampart. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 68–80, November 1994.

[22] R. L. Rivest. *RFC 1321: The MD5 Message Digest Algorithm*. Internet Activities Board, April 1992.

[23] 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, February 1978.

[24] C. P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89 Proceedings* (Lecture Notes in Computer Science 435), pages 239–252. Springer-Verlag, 1990.

[25] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.