The Design and Implementation of a Secure Auction Service

Matthew K. Franklin and Michael K. Reiter

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 fewer than one-third of the servers comprising the auction service. Specifically, it is guaranteed that 1) bids of correct bidders are not revealed until after the bidding period has ended, 2) the auction house collects payment for the winning bid, 3) losing bidders forfeit no money, and 4) only the winning bidder can collect the item bid upon. We also discuss techniques to enable anonymous bidding.

Index Terms—Distributed systems, security, Byzantine failures, electronic commerce, sealed-bid auctions, verifiable signature sharing.

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" [1].

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., [2], [3], [4]), 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 [5].

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 [6]. 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 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 insider 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 fewer than 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

• *M.K. Franklin and M.K. Reiter are with AT&T Research, 600 Mountain Ave., Murray Hill, NJ 07974-0636.*
E-mail: {franklin, reiter}@research.att.com.

Manuscript received Sept. 20, 1995.
Recommended for acceptance by C. Meadows.
For information on obtaining reprints of this article, please send e-mail to: transses@computer.org, and reference IEEECS Log Number S96047.

bidders to submit anonymous bids. The resilience of our service to malicious auction servers can be leveraged to provide resilience to malfeasant auction house insiders. If each individual is allowed access to fewer than one-third of the servers (e.g., through spatial and administrative separation), then corrupting an insider provides no advantage to a bidder in the auction. This reduces the incentive for "buying off" insiders in the auction house.

Our focus in this work is on an efficient and practical approach to performing auctions. We have implemented a prototype of our service to demonstrate its feasibility. The performance of this implementation indicates that our approach is feasible using off-the-shelf workstations for auction servers, even for large auctions involving hundreds of bids. In order to achieve this level of performance, our service employs a range of old and new cryptographic techniques, the secure and efficient integration of which was the primary challenge in this work. The resulting system demonstrates novel and efficient methods for protecting electronic currency in competitive environments. It also provides insights into addressing similar issues in other competitive financial vehicles such as other types of auctions (which have already made an appearance on the Internet [7]), 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 modify our protocol to protect bidder anonymity 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 *insiders* 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 insider 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 insider manipulates the closing time of the bidding period. For example, an insider attempts to prematurely close the bidding period in an effort to exclude some bids.
- Bids for one auction are diverted to a second auction with an earlier closing time, causing their amounts to be revealed prematurely to an insider.
- After the close of the bidding period, a bidder arranges to withdraw a bid or insert a bid, in collaboration with an insider.
- An insider awards the auction item to someone other than the winning bidder (and goes undetected because bids are not made public).
- An insider 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 fewer than one-third 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.

2.1 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.

2.2 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 $\langle m \rangle^{K^{-1}}$. 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 correct 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 [8]. We denote the reliable multicast of message m from a process $S \in \mathcal{G}$ to the group $\mathcal{G}$ by

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

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

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

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 [9].

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

$$B \overset{A}{\rightarrow} \mathcal{G}: \quad m$$

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 nonmember failures) provided that $n \geq 3t + 1$ [9].[1] As described in Section 4, however, this is not the only factor limiting the fault-tolerance of our auction protocol.

3.2 Threshold Secret Sharing Schemes

A (t, n)-threshold secret sharing scheme [10], [11] 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 [11]. In this scheme, the secret s is an element of a finite field F and the ith 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 $t < n$.

As observed by Feldman [12], 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.

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 [9] for details.

3.3 Electronic Money

In its basic form, an electronic money or "digital cash" scheme [2] is a set of cryptographic protocols for

1) a customer to withdraw electronic money from a bank,
2) the customer to use the money to purchase something from a vendor, and
3) 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) [13].

A money scheme is said to be "off-line" [14] 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 $\langle v_\$, \sigma_{\text{bank}}(v_\$), w_\$ \rangle$, where $v_\$$ is a description of the coin, $\sigma_{\text{bank}}(v_\$)$ 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.

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) [15].

4.1 Informal Description of 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 group 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 some of the members are faulty. In addition, faulty members gain no information prior to reconstruction about the signature held by a (correct) sharer.

VΣS has applications whenever a signed document should become valid only under certain conditions (e.g., a will, a "springing power of attorney" [16], or an exchange of contracts). Verifiably sharing the document's signature among a group of processes, with "trigger" instructions, ensures that the signature will not be released until the correct members believe that the triggering events have occurred.

In [15], we develop simple and efficient VΣS schemes for signature schemes based on the discrete logarithm problem, including ElGamal [17], Schnorr [18], and the Digital Signature Algorithm [19]. Sharing requires a single group multicast from the signature holder to the group of processes among which the signature is to be shared, followed by a single round of multicasts among the group members. Reconstruction requires no interaction, beyond a single message sent from each member to the process performing the reconstruction. Our protocols tolerate a faulty sharer and t faulty members in a group of size $n \geq 3t + 1$. Our protocols for ElGamal and Schnorr signatures ensure the secrecy of the signature in a strong sense (related to simulatability); others (e.g., for DSA) provide only a weaker, heuristic notion of secrecy.

Also in [15], we proposed VΣS schemes for exponentiation based signature schemes (RSA [20] and Rabin [21]). However, we later discovered a flaw in our proof of security and in the VΣS schemes themselves [22]. A provably secure but less efficient VΣS scheme for RSA can be obtained from distributed function sharing techniques [23].

4.2 Abstract Description of VΣS

The choice of VΣS scheme for our auction protocol depends on the signature scheme used by the bank to sign its digital cash. 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:

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

Upon receipt of such an atomic multicast, S_i performs a local computation to determine whether the ith private value (which it decrypts with K_i^{-1}) is consistent with the

2. A weaker multicast can be used (see [15]), 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.

public value. S_i reliably multicasts the status of this computation, denoted VΣS-stat$_i$, to the group:

$$S_i \xrightarrow{R} \mathcal{G}: \quad \text{VΣS - stat}_i.$$

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.

4.3 Example of VΣS for Schnorr Signatures

As an example of a VΣS scheme, here we outline the VΣS scheme for Schnorr from [15], which relies on techniques of verifiable *secret* sharing due to Pedersen [24] and Feldman [12]. For a Schnorr signature, the public key is g, p, q, y where p is a large prime, q is a large prime factor of $p-1$, g has order q in Z_p^*, and $y = g^x \bmod p$ for some x. The private key is x. The signature of a document m is given by $\sigma(m) = [w, z]$ where $w = h(g^r \bmod p, m)$ for random $r \in Z_q$ and message digest function h (e.g., MD5 [25]), and where $z = wx + r \bmod q$. A signature can be publicly verified by checking that $w = h(g^z y^{-w} \bmod p, m)$.

To share $\sigma(m)$ to a group $\mathcal{G} = \{S_1, \ldots, S_n\}$, the sharer B chooses values $a_1, \ldots, a_t$ at random from Z_q. Then B lets

$$\text{VΣS-pub}\,(\sigma(m)) = \{w, g^z \bmod p, g^{a_1} \bmod p, \ldots, g^{a_t} \bmod p\},$$

and VΣS-priv$_j$($\sigma(m)$) = $f(j) \bmod q$, where $f(x) = z + a_1 x + \ldots + a_t x^t$. That is, B executes

$$B \xrightarrow{A} \mathcal{G}: \quad m, w, g^z \bmod p, \{g^{a_j} \bmod p\}_{1 \le j \le t}, \left\{\left\langle f(j) \bmod q \right\rangle_{K_j}\right\}_{S_j \in \mathcal{G}}$$

Now suppose that S_i receives an atomic multicast of the form

$$m, w, u, \{u_j\}_{1 \le j \le t}, \{v_j\}_{S_j \in \mathcal{G}}$$

for some values of m, w, u, $\{u_j\}_{1 \le j \le t}$, and $\{v_j\}_{S_j \in \mathcal{G}}$. S_i computes VΣS-stat$_i$ as

$$\text{VΣS - stat}_i = \begin{cases} \text{allow} & \text{if } g^{\langle v_i \rangle^{K_i^{-1}}} = u \prod_{1 \le j \le t} u_j^{i^j} \bmod p \\ \text{complain} & \text{otherwise} \end{cases}$$

S_i reliably multicasts VΣS-stat$_i$ and collects status values from the other servers. Finally, S_i accepts if $w = h(uy^{-w} \bmod p, m)$, $u^q \equiv 1 \bmod p$ and at most t processes complained.

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. Our protocol requires that $n \ge 3t + 1$.

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 $\langle v_\$, \sigma_{\text{bank}}(v_\$), w_\$ \rangle$ 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 $\sigma_{\text{bank}}(v_\$)$ 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 $\sigma_{\text{bank}}(v_\$)$). 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 $\sigma_{\text{bank}}(v_\$)$ 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. 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 [20]). Each K_i is assumed to be available to all servers and bidders; the corresponding private key K_i^{-1} is 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 ith share of s produced via Shamir's (t, n)-threshold secret sharing scheme.

5.1.1 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 $\langle v_\$, \sigma_{\text{bank}}(v_\$), w_\$ \rangle$ 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:

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

5.1.2 Closing the Bidding Period

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

$$\text{(M2)}\quad S_i \xrightarrow{A} \mathcal{G}:\ aid, \textsf{close}$$

When S_i has received (by atomic multicast) close messages for auction *aid* 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 for auction *aid* received prior to closing.

5.1.3 Opening the Bids

Suppose that the lth bid for auction *aid* received (by atomic multicast) at S_i is of the form

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

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 for auction *aid*. These bids are opened in three steps:

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

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

S_i then executes:

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

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 $\left\langle s_{j,l} \| aid\right\rangle_{K_j} = c_{j,l}$. 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 lth 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 $s_{j_1,l} \ldots s_{j_{t+1},l}$. If $\left\langle f_l(j) \| aid\right\rangle_{K_j} \neq c_{j,l}$ for any j, $1 \le j \le n$, then S_i discards the lth bid. S_i also discards the lth 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 lth bid is discarded, then it was submitted by a faulty bidder.

5.1.4 Checking the Validity of Bids

S_i checks the validity of the remaining bids as follows.

1) For each remaining bid, S_i first performs the validity checks on $v_\$$ and $w_\$$ 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 $\left\langle \text{priv}_{i,l}\right\rangle^{K_i^{-1}}$ and pub_l; see (*)) for each l, $1 \le l \le L'$, according to the VΣS scheme, and executes

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

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.

5.1.5 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*. Such messages from $t + 1$ servers can collectively serve as B's ticket for claiming the auctioned item.

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. The procedure by which this coin is reconstructed and deposited with a bank is outside the scope of our protocol. However, we caution against the servers reconstructing 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, 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. Since the reconstruction and deposit of the coin can occur at any time after the auction (e.g., at the end of the day), a range of manual and/or electronic procedures are possible for performing these operations safely. A particularly convenient solution would be for each server to forward its private VΣS value for that coin's signature to the bank, so that the bank can perform the reconstruction itself.

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.

5.2.1 Eliminating VΣS

It is possible to eliminate the use of a VΣS scheme by having the bidder share $\sigma_{bank}(v_\$)$ among the servers with a standard threshold secret sharing scheme or a *verifiable* secret sharing scheme [26]. 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 faulty 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).

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 [5]). 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.

5.2.2 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., [2], [3]).

5.2.3 Promisory Bidding

Rather than requiring bidders to submit bids containing digital cash in the full amount of the bid, the service could accept bids containing cash for only a portion of the bid amount. The bid would serve as a promise to complete the payment if that bid wins, and the cash portion (if any) would serve as a "good faith" deposit. This alternative may be preferable in auctions drawing large bids. However, it offers the opportunity for a winning bidder to default on its payment after all other bids have been opened. In addition, it complicates collection of the winning bid, requiring protocols to collect that payment and to determine a new winner in case the original winning bidder defaults. As above, the message complexity of determining the actual winner could then be proportional to the number of uncollectable bids submitted.

5.2.4 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 [27] 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 could expose the service to substantial computational overheads induced by faulty bidders submitting large numbers of bids. Such an attack would be less effective against our protocol, because to open bids, the main costs per server per bid are polynomial interpolations and (re-)encryptions, which are relatively inexpensive with an appropriate choice of encryption algorithm (e.g., RSA with reasonably small encrypting exponent).

A threshold signature scheme [27], 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.

5.2.5 Mental Games

"Mental games" [28] 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.

5.2.6 Nonmalleable Cryptography

Intuitively, an encryption scheme is "nonmalleable" [29] if it is infeasible to modify a ciphertext so that a known relationship will hold between the new plaintext and the

original plaintext. One of the motivating examples for nonmalleable encryption was contract bidding, where an attacker might try to become the low bidder by manipulating all competitors' bids upward. In our scheme, the inclusion of digital cash makes this particular attack irrelevant. However, malleability could lead to other weaknesses, e.g., that enable an attacker to divert a bid to an auction with an earlier closing time. Our auction protocol precludes the use of non-malleable encryption by exploiting determinism in the servers' encryption schemes (steps 2 and 3 of "opening the bids"); the theoretical definition of nonmalleability requires that encryption be probabilistic. Moreover, existing encryption schemes that are provably nonmalleable would be prohibitively inefficient for our purposes.

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 rigorous proof of security.

6.1 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.*

   Due to the properties of atomic multicast, all correct servers agree on which bids were received before the close of the bidding period. If any correct server S_i sends $s_{i,l} = \perp$, then the lth bid will be discarded by all correct servers, because each correct server will either receive too many $\perp$ values to determine an f_l or will notice that $\langle f_l(i) \| aid \rangle_{K_i} \neq c_{i,l}$. If $s_{i,l} \neq \perp$ for each correct S_i, then either each correct server will determine the same f_l or each correct server will detect discrepancies between its f_l and the values $\{c_{j,l}\}_{S_j \in \mathcal{G}}$. Thus, all correct servers agree on the bids remaining after "opening the bids." Moreover, these include all well-formed bids, since any bad $s_{i,l} \neq \perp$ provided by a faulty S_i is discarded in step 2 of "opening the bids."

   From the remaining bids, all correct servers agree on the subset that pass the validity checks, by the properties of the digital cash and VΣS schemes. All correct servers select the same winning bid from these acceptable bids, by following the public rule for determining the winner. Finally, all correct servers 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.

6.2 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 the VΣS implementations proposed in [15], the public and private VΣS information available to the coalition would yield at most a message digest dependent on $v_\$$. 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, this message digest reveals no useful information about the amount of the bid. Lastly, 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, and the bid will be rejected before the amount or source of the bid are revealed.

7 Performance

We have implemented a research 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 [9], and

employs CryptoLib [30] 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 $B \parallel v_\$ \parallel w_\$$ when submitting a bid, we share a (much smaller) key to a symmetric cipher (specifically, DES [31]) and include in the bid the encryption of $B \parallel v_\$ \parallel w_\$$ under that key. In addition, since each server must receive only $t + 1$ shares of this key to recover $B \parallel v_\$ \parallel w_\$$, 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.

Approximate latency numbers in milliseconds for the stages of the auction protocol in the case of no failures are shown in Table 1 and Fig. 1. These numbers were derived from tests on a network of moderately loaded SPARCstation 10s. These tests used RSA public key encryption with 512-bit moduli and 8-bit encryption exponents, and Schnorr signatures with 512-bit modulus p and 160-bit prime factor q of $p - 1$. As a result, the VΣS scheme of [15] based on Schnorr signatures (see Section 4) was also used. There were four auction servers, which is the minimum number of servers required to tolerate the failure of one auction server (i.e., $t = 1$).

TABLE 1
OPERATIONS WITH CONSTANT LATENCY

Operation	Latency (ms)
Submitting a bid	327
Closing the bidding period	87
Declaring the winner	94

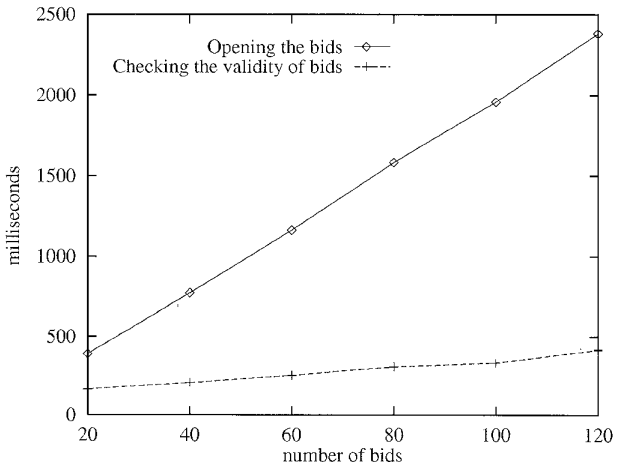

Fig. 1. Latencies that grow with number of bids.

In order to isolate the costs of our auction protocol, the numbers in Table 1 and Fig. 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 Fig. 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., [32] using a 512-bit modulus).

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 the bidder creating a bid and atomically multicasting it to the server group, and each server S_i decrypting the two portions of the message private to it and performing as many local computations as possible associated with the VΣS scheme. These decryptions and VΣS operations consume just over 200 milliseconds at each server, which implies that the service can process at most five bids per second during the bidding period. "Closing the bidding period" includes the latency of $2t + 1$ servers initiating atomic multicasts (close messages) in parallel, and each waiting to receive $t + 1$ of those messages. "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.

Fig. 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 $B \parallel v_\$ \parallel w_\$$ 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" includes the latency of each server, in parallel, reliably multicasting its VΣS-stat values and completing the verification for each bid, until each is either accepted or rejected. Note that "checking the validity of bids" reflects only those local VΣS computations that depend on $v_\$$ values. In our implementation, all other VΣS computations are performed immediately when the bid is received, and are reflected in "submitting a bid" of Table 1.

We reiterate that the latencies in Table 1 and Fig. 1 are approximate, due to the difficulty of precisely measuring distributed events. The latencies of the stages involving only the auction servers (i.e., closing the bidding period, opening the bids, and checking the validity of bids) were computed as the average of the latencies of these stages as measured at each server individually. In order to measure the latency of submitting a bid, we modified each auction server to reply with a point-to-point message to the bidder after processing the bid. (Obviously, bidder atomic multicasts were not anonymous, as assumed in Section 3.1.) The latency of submitting the bid was then measured as the elapsed time at the bidder between initiating a bid and receiving replies from all servers. Finally, the latency of declaring a winner is simply a sum of the measured latencies of a Schnorr signature, $t + 1$ Schnorr verifications, and message transport.

The factor limiting performance in each stage of our auction protocol is cryptographic operations. This is even true for those stages involving little computation beyond that involved in reliable or atomic multicasts, as the latencies of the multicast protocols we used [9] are also dominated by cryptography. These performance numbers are thus very sensitive to choices of cryptographic algorithms and key lengths. Moreover, they should improve substantially if more powerful server machines are used.

8 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.

A first 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

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

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. The winner can employ $t + 1$ such declarations and the number r, which only it knows, as its ticket for claiming the auctioned item.

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 $\sigma_{\mathrm{bank}}(v_\$)$ 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.

9 Conclusion

We have presented the design and implementation of a practical distributed auction service that can tolerate the malicious behavior of fewer than one-third 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.

Acknowledgments

Matthew K. Franklin would like to thank Stuart Haber for early discussions on cryptographic auctions. We also thank the anonymous referees for helpful suggestions.

References

[1] W. Diffie, "The Impact of a Secret Cryptographic Standard on Encryption, Privacy, Law Enforcement and Technology," Hearing before the Subcommittee on Energy and Commerce, U.S. House of Representatives, 103rd Congress, First Session, Apr. 29 and June 9, 1993, serial no. 103-53, pp. 111-116.

[2] D. Chaum, "Security without Identification: Transaction Systems to Make Big Brother Obsolete," *Comm. ACM*, vol. 28, pp. 1,030-1,044, 1985.

[3] G. Medvinsky and B.C. Neuman, "NetCash: A Design for Practical Electronic Currency on the Internet," *Proc. First ACM Conf. Computer and Comm. Security*, pp. 102-106, Nov. 1993.

[4] J. Boly et al., "The ESPRIT Project CAFÉ—High Security Digital Payment System," *Computer Security—ESORICS 94*, D.Gollmann, ed., *Lecture Notes in Computer Science 875*, Springer-Verlag, 1994.

[5] R. McAfee and J. McMillan, "Auctions and Bidding," *J. Economic Literature*, vol. 25, pp. 699-738, June 1987.

[6] "FCC Takes Licenses, Denies More Time to 13 Bidders," AP-Dow Jones News, Aug. 10, 1994.

[7] P.H. Lewis, "Auction of Collectibles on the Internet," *New York Times*, May 23, 1995.

[8] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Trans. Programming Languages and Systems*, vol. 4, no. 3, pp. 382-401, July 1982.

[9] M.K. Reiter, "Secure Agreement Protocols: Reliable and Atomic Group Multicast in Rampart," *Proc. Second ACM Conf. Computer and Comm. Security*, pp. 68-80, Nov. 1994.

[10] G.R. Blakely, "Safeguarding Cryptographic Keys," *Proc. AFIPS Nat'l Computer Conf.*, pp. 313-317, 1979.

[11] A. Shamir, "How to Share a Secret," *Comm. ACM*, vol. 22, no. 11, pp. 612-613, Nov. 1979.

[12] P. Feldman, "A Practical Scheme for Non-Interactive Verifiable Secret Sharing," *Proc. 28th IEEE Symp. Foundations of Computer Science*, pp. 427-437, Oct. 1987.

[13] S. Even, O. Goldreich, and Y. Yacobi, "Electronic Wallet," *Proc. Crypto '83*, Plenum Press, 1984.
[14] D. Chaum, A. Fiat, and M. Naor, "Untraceable Electronic Cash," *Advances in Cryptology—Crypto '88 Proc.*, S. Goldwasser, ed., *Lecture Notes in Computer Science 403*, pp. 319-327, Springer-Verlag, 1989.
[15] M.K. Franklin and M.K. Reiter, "Verifiable Signature Sharing," *Advances in Cryptology—Eurocrypt '95*, L.C. Guillou and J. Quisquater, eds., *Lecture Notes in Computer Science 921*, pp. 50-63, Springer-Verlag, 1995.
[16] J. Hoffman, "New Power-of-Attorney Form Is Introduced," *New York Times*, Oct. 1, 1994.
[17] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Trans. Information Theory*, vol. 31, no. 4, pp. 469-472, July 1985.
[18] C.P. Schnorr, "Efficient Identification and Signatures for Smart Cards," *Advances in Cryptology—Crypto '89 Proc.*, G. Brassard, ed., *Lecture Notes in Computer Science 435*, pp. 239-252, Springer-Verlag, 1990.
[19] "NIST FIPS PUB 181, Digital Signature Standard," U.S. Dept. of Commerce/National Inst. of Standards and Technology.
[20] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM*, vol. 21, no. 2, pp. 120-126, Feb. 1978.
[21] M.O. Rabin, "Digitalized Signatures and Public Key Functions as Intractable as Factorization," Technical Report MIT/LCS/TR-212, Laboratory for Computer Science, Massachusetts Inst. of Technology, Jan. 1979.
[22] D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter, "Low-Exponent RSA with Related Messages," *Advances in Cryptology—Eurocrypt '96*, Springer-Verlag, to appear.
[23] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung, "How to Share a Function Securely," *Proc. 26th ACM Symp. Theory of Computing*, pp. 522-533, May 1994.
[24] T.P. Pedersen, "Distributed Provers with Applications to Undeniable Signatures," *Advances in Cryptology—Eurocrypt '91 Proc.*, D.W. Davies, ed., *Lecture Notes in Computer Science 547*, pp. 221-242, Springer-Verlag, 1991.
[25] R.L. Rivest, *RFC 1321: The MD5 Message Digest Algorithm*, Internet Activities Board, Apr. 1992.
[26] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults," *Proc. 26th IEEE Symp. Foundations of Computer Science*, pp. 383-395, Oct. 1985.
[27] Y. Desmedt, "Threshold Cryptography, "*European Trans. Telecommunications and Related Technologies*, vol. 5, no. 4, pp. 449-457, July 1994.
[28] O. Goldreich, S. Micali, and A. Wigderson, "How to Play Any Mental Game," *Proc. 19th ACM Symp. Theory of Computing*, pp. 218-229, May 1987.
[29] D. Dolev, C. Dwork, and M. Naor, "Non-Malleable Cryptography," *Proc. 23rd ACM Symp. Theory of Computing*, pp. 542-552, May 1991.
[30] J.B. Lacy, D.P. Mitchell, and W.M. Schell, "CryptoLib: Cryptography in Software," *Proc. Fourth USENIX Security Workshop*, pp. 1-17, Oct. 1993.
[31] "Data Encryption Standard," Nat'l Bureau of Standard, Federal Information Processing Standards Publication 46, Government Printing Office, Washington, D.C., 1977.
[32] S. Brands, "Untraceable Off-Line Cash in Wallets with Observers," *Advances in Cryptology—Crypto '93*, D.R. Stinson, ed., *Lecture Notes in Computer Science 773*, pp. 302-318, Springer-Verlag, 1994.

Matthew K. Franklin received the BA degree in mathematics from Pomona College in 1983, the MA degree in mathematics from the University of California, Berkeley, in 1985, and the PhD degree in computer science from Columbia University in 1994. He is currently a principal investigator/member of the technical staff at AT&T Research (formerly AT&T Bell Laboratories). His research interests include cryptography and distributed computing.

Michael K. Reiter received the BS degree in mathematical sciences from the University of North Carolina at Chapel Hill in 1989, and the MS and PhD degrees in computer science from Cornell University in 1991 and 1993, respectively. He is currently a principal investigator/member of the technical staff at AT&T Research (formerly AT&T Bell Laboratories). His research interests include security and fault-tolerance in distributed computer systems.