Efficient Private Bidding and Auctions with an Oblivious Third Party*

Christian Cachin

IBM Research Division
Zurich Research Laboratory
CH-8803 Rüschlikon, Switzerland
cca@zurich.ibm.com

July 12, 1999

Abstract

We describe a novel and efficient protocol for the following problem: A wants to buy some good from B if the price is less than a. B would like to sell, but only for more than b, and neither of them wants to reveal the secret bounds. Will the deal take place? Our solution uses an oblivious third party T who learns no information about a or b, not even whether $a > b$. The protocol needs only a single round of interaction, ensures fairness, and is not based on general circuit evaluation techniques. It uses a novel construction, which combines homomorphic encryption with the Φ-hiding assumption and which may be of independent interest. Applications include bargaining between two parties and secure and efficient auctions in the absence of a fully trusted auction service.

Keywords: Bidding, Auctions, Homomorphic cryptosystems, Φ-Hiding assumption, Private information retrieval, Multiparty computation.

1 Introduction

Suppose A wants to buy some good from B if the price is less than a. B would like to sell, but only for more than b, and neither of them wants to reveal the secret bounds a and b. Will the deal take place? We call this the problem of *private bidding*: Ideally, A and B have a trusted device at hand, in which they privately enter a and b. The device outputs either "yes" if $a > b$ or "no," but no further information. Only if the result is "yes" does A have to reveal a and the deal takes place at this price (other policies could be applied here). This work presents a fair and efficient protocol for private bidding that uses a partially trusted third party T, who learns no information about a or b.

An efficient solution for private bidding enables new forms of electronic business transactions that are impossible in face-to-face situations. The recent success of Web auction services shows that person-to-person auctions are an important concept in online trading. These services do not (yet) offer privacy to the bidders, although privacy is becoming more and more important in the online world. Private bidding is also reminiscent of bargaining, once a very popular method to determine a price and still used in many places.

The technical aspects of private bidding are by no means new to cryptography. In fact, the problem was introduced almost twenty years ago as the "millionaire's problem" [Yao82]: Two parties want to determine who is richer without disclosing anything else about their wealth. A

*This is IBM Research Report RZ 3131 (May 25, 1999).

number of early solutions for this problem have become part of the folklore. None of them is efficient, though, because for l-bit numbers a and b, the work is in $\Theta(2^l)$.

The development of techniques for general secure multiparty computation subsequently showed that any two- or multi-party function can be computed securely [GMW87, BGW88, CCD88, AF90, Gol98]. Thus, A and B can perform private bidding efficiently even without help from a third party! However, the generic two-party constructions do not ensure fairness in an efficient way; and if a third party is involved for this purpose, it is unclear how to maintain the privacy of the losing bid.

In contrast, our protocol is fair and efficient: it requires only two messages between bidders A and B and two messages between each bidder and T. The protocol does not use the general method based on circuits and gains its efficiency through the help of the semi-trusted party T. A and B trust T to perform some service for them, but they do not want T to learn anything about their bids. Thus, T can misbehave on its own and try to get information about a or b, but it does not collude with either A or B.

Our solution is based on a combination of homomorphic public-key encryption and a number-theoretic construction relying on the ϕ-hiding assumption, which was first used for private information retrieval (PIR) schemes with low communication [CMS99]. This construction may be of independent interest.

The protocol guarantees *fairness* for A and B in the following sense: A learns the result of $a > b$ if and only if B learns it. Without help from a neutral party, ensuring fairness between distrusting parties is a notoriously time-consuming task in the digital world and typically requires many rounds of interaction (e.g. [BGMR90]). Recent work on fair exchange (see the references in Sect. 1.3) shows how a third party can provide fairness in practical protocols, even if it is not involved in regular transactions and only used if needed. Fairness is ensured here because both parties commit to their inputs. The winner has to reveal its number and the other party can verify that this was the number used in the protocol without disclosing the losing bid.

1.1 Overview of the Protocol

We give a short description here; the complete protocol can be found in Section 3. Let $a, b \in [0, 2^l - 1]$ be A's and B's inputs, respectively. The idea behind the protocol is that T blindly compares the bits of a and b, starting with the most significant bit. At the first index where the bits differ, a result flag is registered (invisibly for T) that determines whether $a > b$. The result is retrieved by A and B using the mechanism of the PIR protocol based on the Φ-hiding assumption.

We say that a modulus $m = p'q'$ *hides* a prime p if p' and q' are large (e.g. 500-bit) primes such that $p' = 2q_1 + 1$ with q_1 prime and $p' = 2pq_2 + 1$, where p, q_2 are odd primes and p is short (e.g. 100 bits). Thus, p divides $\phi(m)$. The Φ-hiding assumption (ΦHA) is: for a randomly chosen m that hides a prime p_0 and an independent, randomly chosen short prime p_1, it is hard to distinguish whether p_0 or p_1 is a factor of $\phi(m)$. Given a number m that hides p, a list of short primes $p_1, \ldots, p_n$, and a starting value $g \in \mathbb{Z}_m$ (that has no pth roots modulo m), suppose one raises g to the power $\prod_i p_i$ modulo m. The result has a pth root modulo m if and only if $p = p_i$ for some p_i in the list; however, determining this or finding i is assumed to be hard without knowledge of m's factorization (see Section 2.3).

Let $\mathcal{S}$ be a semantically secure public-key system with (probabilistic) encryption function $E_T : \mathcal{X} \to \mathcal{C}$ and decryption function $D_T : \mathcal{C} \to \mathcal{X}$ that satisfies the following homomorphic property. Let $(\mathcal{X}, +, 0)$ denote an Abelian group on messages and $(\mathcal{C}, \cdot, 1)$ an Abelian group on ciphertexts; for $x_0, x_1 \in \mathcal{X}$, it holds $E_T(x_0 + x_1) = E_T(x_0) \cdot E_T(x_1)$. ($\mathcal{S}$ also has to satisfy $D_T(E_T(x_0 + x_1)) \neq D_T(E_T(x_0 - x_1))$, see Section 2.4.) At the beginning of the protocol, T chooses a public-key/secret-key pair for $\mathcal{S}$, publishes E_T and keeps D_T secret.

A and B choose random $x_l, x_{l-1}, \ldots, x_0 \in \mathcal{X}$ and $s_{l-1}, \ldots, s_0 \in \mathcal{X}$. A chooses a random modulus m_A that hides a short prime p_A; similarly B chooses m_B that hides p_B. Let λ be a one-way map from $\mathcal{X}$ to primes of the same length as p_A, p_B such that $\lambda(x_j + s_j) = p_A$ and $\lambda(x_j - s_j) = p_B$ for $j = 1, \ldots, l$. A prepares

$$y_{A,j} \;=\; E_T(x_j - x_{j+1} + a_j s_j)$$

(where $a_{l-1}, \ldots, a_0$ are the bits of a) and B prepares

$$y_{B,j} \;=\; E_T(-b_j s_j)$$

(where multiplication of $s_j \in \mathcal{X}$ by a bit a_j is interpreted in the natural way). Together, they compute $z_j \;=\; y_{A,j} \cdot y_{B,j}$ for $j = l-1, \ldots, 0$ and send $x_l, z_{l-1}, \ldots, z_0$ and m_A, m_B to T.

T computes $c_l = x_l$, chooses $g_{A,l} \in \mathbb{Z}_{m_A}$ and $g_{B,l} \in \mathbb{Z}_{m_B}$ at random and repeats for $j = l-1, \ldots, 0$:

1. $c_j = c_{j+1} + D_T(z_j)$;
2. $g_{A,j} = (g_{A,j+1})^{\lambda(c_j)} \mod m_A$;
3. $g_{B,j} = (g_{B,j+1})^{\lambda(c_j)} \mod m_B$;

T sends $g_{A,0}$ to A and $g_{B,0}$ to B. The bidders can determine whether $a > b$ by checking whether $g_{A,0}$ has a p_Ath root modulo m_A using the factorization of m_A (and similarly for B). Note that during T's loop on $j = l-1, \ldots$, it holds that $c_j = x_j$ as long as $a_j = b_j$, then $\lambda(c_j)$ is equal to p_A or p_B once, but is different from p_A and p_B afterwards.

1.2 Applications

Apart from two-party private bidding by itself, our protocol can be used for bargaining on the Internet and as a building block for secure auction protocols.

The Internet is already a popular place to conduct auctions, as can be seen from the literature (e.g. [BS98, KF98]), but even more so from the recent rise of companies offering online auctions such as eBay and Onsale.[1]

Bargaining is an ancient and still very popular method to determine a price, and our protocol enables efficient bargaining in the digital world. The private bidding protocol is used repeatedly until the price is fixed. A starts by offering a (too) low amount a and B initially asks a (too) high price b. They repeat the protocol with changing values until $a > b$ and A (or B, depending on the policy) has to reveal the closing price. In contrast to face-to-face bargaining, no party learns anything about the strategy of the other party.

The private bidding protocol forms a building block for sealed-bid auctions with a partially trusted auction service. Ideally, the bidders want their bids to remain secret unless they win the auction (and obviously have to announce their bid). In Section 4, we describe briefly a practical protocol for cryptographically protected sealed-bid auctions with only two auction servers. The two semi-trusted servers are assumed not to collude. One server learns some information about the bids (namely their partial order) and the other server learns no information at all. Previous protocols were based either on general multiparty computation techniques [HTK98] or ensured primarily fairness, but not secrecy [FR96].

[1] A search on April 26, 1999 for "auctions" in `news.com` turned up 52 news items in April 1999 alone.

1.3 Related Work

A semi-trusted third party was first used for efficient fair exchange protocols by Franklin and Reiter [FR97]. The third party has to be present for the transaction, but it is prevented from learning anything about the messages exchanged by a cryptographic protocol.

The *optimistic* approach to fair exchange by Asokan, Schunter, Shoup, and Waidner [ASW97, ASW98] does not need the third party during regular operation of the system; T only handles exceptions, such as network failures or attempts to cheat. A similar approach for delivering certified email with an "invisible" third party was independently presented by Micali [Mic97].

A protocol for secure sealed-bid online auctions has been proposed by Franklin and Reiter [FR96]. Their auction service consists of s servers of which up to $t \leq \lfloor \frac{s-1}{3} \rfloor$ can be corrupted. It ensures fairness for the bidders, guarantees that payment can be collected from the winner, and hides the bids, but only until the bidding period closes. The bids are later opened collaboratively by the servers and the winner is determined in the open, so that all servers can see the bids.

The recent work of Harkavy *et al.* [HTK98] describes an auction service for secure sealed-bid auctions, in which only the winning bid is disclosed. It is based on general techniques for secure multiparty computation and can tolerate up to $t \leq \lfloor \frac{s-1}{3} \rfloor$ corrupted servers. The protocols are practical only for small values of s.

2 Tools

2.1 Preliminaries

The security parameter is denoted by k and we use $k', k'', \ldots$ to denote additional security parameters. Wherever $k'', \ldots$ occurs, it is implicitly assumed that there is a polynomial $p'(\cdot)$ and $k' = \Theta(p'(k))$. For $k'', \ldots$ the assumption is analogous. The notation $[a, b]$ denotes the interval $\{a, \ldots, b\} \subset \mathbb{Z}$. For an l-bit number a, let $a_{l-1}, \ldots, a_0$ be its binary representation such that $a = \sum_{j=0}^{l-1} a_i 2^j$. The binary representation of $a_r \in [0, 2^l - 1]$ is denoted by $a_{r,l-1}, \ldots, a_{r,0}$. The concatenation of strings is denoted by $\|$.

For a positive integer m, let QR_m denote the subgroup of squares in $\mathbb{Z}_m^*$. The *Euler totient function* of an integer m, denoted by $\phi(m)$, is defined as the number of positive integers $\leq m$ that are relatively prime to m.

An *algorithm* is a (probabilistic) Turing machine. A *probabilistic polynomial-time* (PPT) algorithm runs in polynomial time except for an exponentially small fraction of its random choices.

The statistical difference between two probability distributions P_X and P_Y is denoted by $|P_X - P_Y|$. A quantity ϵ_k is called *negligible* (as a function of k) if for all $c > 0$ there exists a constant k_0 such that $\epsilon_k < \frac{1}{k^c}$ for all $k > k_0$; otherwise, it is called *noticeable* or non-negligible.

The formal security notion is defined in terms of indistinguishability of probability ensembles indexed by k, but extension from a single random variable to an ensemble is usually assumed implicitly.

Two probability ensembles $X = \{X_k\}$ and $Y = \{Y_k\}$ are called *statistically indistinguishable* (written $X \stackrel{s}{\approx} Y$) if $|P_{X_k} - P_{Y_k}|$ is negligible.

Two probability ensembles $X = \{X_k\}$ and $Y = \{Y_k\}$ are called *computationally indistinguishable* (written $X \stackrel{c}{\approx} Y$) if for every algorithm D that runs in probabilistic polynomial time (in the length of its inputs), $|\mathrm{P}[D(1^k, X_k) = 1] - \mathrm{P}[D(1^k, Y_k) = 1]|$ is negligible.

We also need a hash function family $\mathcal{H}$ with the following property that generalizes *universal one-way hash functions* [NY89]: $\mathcal{H}$ consists of efficiently computable functions $H_k : \mathcal{K}_k \times \{0,1\}^* \to \{0,1\}^{k'}$ such that for all PPT algorithms D, all input strings x of polynomial length in k, and polynomially (in k) many strings $\Delta_1, \ldots, \Delta_{k''} \in \{0,1\}^{k'}$, the probability for a randomly

chosen $\kappa \in \mathcal{K}_k$ that D outputs x' such that $H_k(\kappa, x) = H_k(\kappa, x') \oplus \Delta_i$ for some i is negligible. We call such $\mathcal{H}$ a Δ-*universal one-way hash function.*

A commitment scheme is a protocol between two parties S and R that consists of a commit phase, in which S commits to some secret value, and of an open phase, in which S reveals the commitment and R verifies that this is indeed the value committed to in the first phase. A *non-interactive string commitment scheme* consists of two families of deterministic algorithms, $\{Com_k\}$ and $\{Ver_k\}$. $Com_k(\rho, \sigma)$ takes as inputs two binary strings and outputs a commitment string γ. $Ver_k(\gamma, \rho, \sigma)$ takes as inputs three binary strings and outputs 0 or 1. All strings are of length polynomial in k. The following properties have to hold:

1. for all σ and ρ, $Ver_k(Com_k(\rho, \sigma), \rho, \sigma) = 1$;
2. for two arbitrary σ_0, σ_1 and randomly chosen ρ_0, ρ_1, the commitments $\gamma_0 = Com_k(\rho_0, \sigma_0)$ and $\gamma_1 = Com_k(\rho_1, \sigma_1)$ are computationally indistinguishable;
3. the probability that any PPT algorithm outputs γ, ρ, σ, ρ', σ' such that $Ver_k(\gamma, \rho, \sigma) = 1$ and $Ver_k(\gamma, \rho', \sigma') = 1$ is negligible.

2.2 Multiparty Computation Model

The formal description of our protocol is given in the model of multiparty computation. A multiparty computation (for n parties) is specified by a (possibly randomized) function $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$, where P_i holds private input x_i and wishes to obtain the ith component of $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$. The goal of a multiparty protocol is to emulate an ideal situation, in which a universally trusted party U privately receives $x_1, \ldots, x_n$, evaluates f and sends back y_i to P_i privately (our oblivious T is a regular party P_j in this model).

We distinguish between passive and active cheating. A *passively* cheating party (also called *semi-honest*) follows the protocol, but keeps all records internally. A protocol for computing f is secure against *passive cheating* if whatever the semi-honest parties can compute after participating in the real protocol could essentially be concluded in the ideal situation from their private inputs and outputs of f (if f is evaluated by U).

Actively cheating parties, on the other hand, may execute arbitrary code. Security is again defined relative to what is achievable in the ideal model: A protocol is *robust* (or *secure against active cheating*) if for every cheating strategy, there exists a strategy in the ideal model that achieves essentially the same. In other words, whatever the corrupted parties can do or compute while participating in the protocol, they could also achieve in the ideal situation.

Formalizing general multiparty computation is in itself a nontrivial task with its own history, for which we refer to [Bea91, MR92, Gol98, Can98]. The model used here is given in Section 3.1.

2.3 Φ-Hiding Assumption

Our construction is based on the Φ-*hiding assumption* (ΦHA) [CMS99], which is related to the difficulty of factoring and to the higher-residuosity assumption.

Informally, the ΦHA states that it is computationally intractable to decide whether a given small prime divides $\phi(m)$, where m is a composite integer of unknown factorization. (Recall that computing $\phi(m)$ on input m is as difficult as factoring m.) We also need to assume that it is possible to find efficiently a random composite m such that a given prime p divides $\phi(m)$, but we omit this point for simplicity.

Let $\mathcal{P}_{k'}$ denote the set of all k'-bit primes. Consider the set $\mathcal{R}_{k''}$ that consists of all numbers $m = p'q'$ such that p' and q' are k''-bit primes, one of which is a safe prime (a prime of the form $2q_1 + 1$ with q_1 prime) and the other one is a quasi-safe prime of the form $2pq_2 + 1$, where p, q_2

are odd primes and the length of p is k' bits. We say that a composite $m \in \mathcal{R}_{k''}$ *hides* a prime $p \in \mathcal{P}_{k'}$ if $p|\phi(m)$.

The ΦHA asserts that for a random $m \in \mathcal{R}_{k''}$ that hides a prime $p_0 \in \mathcal{P}_{k'}$ and an independent random prime $p_1 \in \mathcal{P}_{k'}$, the tuples (m, p_0) and (m, p_1) are computationally indistinguishable.

The following fact is useful in connection with the ΦHA.

Lemma 1. *Let $m \in \mathcal{R}_{k''}$ such that m hides a k'-bit prime p and let $g \in \mathbb{Z}_m^*$ be of order $\phi(m)/2$. Choose $r \in \mathbb{Z}_m^*$ randomly with uniform distribution. Then for all numbers e_0, e_1 that are relatively prime to $\phi(m)/2$,*

$$g^{e_0 r} \stackrel{s}{\approx} g^{e_1 r}$$

and

$$g^{pe_0 r} \stackrel{s}{\approx} g^{pe_1 r}.$$

Proof. Because r is chosen randomly in $\mathbb{Z}_{\phi(m)}$, the distribution of $r \mod \phi(m)/2$ is statistically indistinguishable from the uniform distribution over $\mathbb{Z}_{\phi(m)/2}$. As e_0 and e_1 are relatively prime to $\phi(m)/2$, the distribution of $r \cdot e_0/e_1 \mod \phi(m)/2$ is also statistically indistinguishable from the uniform distribution over $\mathbb{Z}_{\phi(m)/2}$, which proves the first statement. The second statement follows analogously. □

2.4 Homomorphic Encryption

We need a public-key cryptosystem $\mathcal{S}$ with semantic security and a homomorphic property. Let k be its security parameter, let $E_k : \{0,1\}^k \times \mathcal{X} \to \mathcal{C}$ denote the public encryption function for a message x using a random string u, and let $D_k : \mathcal{C} \to \mathcal{X}$ be the corresponding secret decryption function.

Semantic security asserts that an eavesdropper cannot get partial information about the plaintext from a cryptogram [GM84]. More precisely, $\mathcal{S}$ is semantically secure if for two arbitrary messages x_0 and x_1 and randomly chosen $u_0, u_1 \in \{0,1\}^k$, the encryptions $E_k(u_0, x_0)$ and $E_k(u_1, x_1)$ are computationally indistinguishable.

The required homomorphic property is that $E_k(u, \cdot)$ is a group homomorphism for all u. Specifically, let $(\mathcal{X}, +, 0)$ be an Abelian group on messages (with inverse operation denoted by $-$) and let $(\mathcal{C}, \cdot, 1)$ be an Abelian group on ciphertexts. Then for all $x_0, x_1 \in \mathcal{X}$ and $u_0, u_1 \in \{0,1\}^k$, there exists u such that $E_k(u, x_0 + x_1) = E_k(u_0, x_0) \cdot E_k(u_1, x_1)$.

We need $|\mathcal{X}| > 2^{k'}$ for some k' and the group operation $+$ has to be different from addition of binary vectors. It should be the case that for randomly chosen $x, y \in \mathcal{X}$, it holds that $x+y \neq x-y$ except with negligible probability. These conditions on $\mathcal{S}$ rule out using Goldwasser-Micali encryption [GM84] based on quadratic residuosity repeated k' times. However, there are alternatives if one switches to higher residues: higher-residuosity encryption as proposed by Benaloh [Ben94] and the higher-residues public-key system by Naccache and Stern [NS98]. Both are reviewed in Appendix A.

3 Two-Party Private Bidding

A two-party bidding protocol with an oblivious third party is a protocol for two users, A and B, with secret inputs a and b, respectively, and a third party T to determine whether $a > b$.

3.1 Model

A, B, and T are interactive, probabilistic Turing machines that communicate pairwise through secure channels (which could be realized by standard cryptographic techniques).

The function to compute is given by the function $f : \mathcal{A} \times \mathcal{B} \times \mathcal{T} \to \{0,1\} \times \{0,1\} \times \mathcal{T}$, where $\mathcal{A} = \mathcal{B} = [0, 2^l - 1]$, $\mathcal{T} = \{\varepsilon\}$, and

$$f(a,b,\varepsilon) \;=\; \begin{cases} (1,0,\varepsilon) & \text{if } a > b \\ (0,0,\varepsilon) & \text{if } a = b \\ (0,1,\varepsilon) & \text{if } a < b. \end{cases}$$

A's and B's inputs a and b are positive l-bit numbers; T's input and output space $\mathcal{T}$ contains only the empty word ε, corresponding to T's role as an oblivious third party. For simplicity, we assume that l and the security parameters k, k', k'' are implicit inputs to all parties and adversaries. (Typically, l and the security parameter k are polynomially related, but l could in fact be bigger; in this case the protocols run in time polynomial in k and l.)

In the two-party bidding protocol, at most one party is assumed to be cheating, either A or B actively, or T only passively.

We first describe the *passive* case (see Section 2.2). The passive *real* adversary is a PPT algorithm C that has access to the internal view of A, B, or T. At the end of the real protocol, the adversary outputs an arbitrary function of its view. Slightly abusing notation, we let the random variables $A(a), B(b)$, and $T(\varepsilon)$ denote the parties' outputs for inputs $a \in \mathcal{A}$, $b \in \mathcal{B}$, and ε, and C the adversary's output.

The passive adversary in the *ideal* process is a PPT algorithm $\overline{C}$ that has access to the view of A, B, or T. The parties give their inputs $(i_A, i_B, i_T) = (a, b, \varepsilon)$ to U (passive), which computes $(o_A, o_B, o_T) = f(i_A, i_B, i_T)$ and sends back the corresponding outputs. All parties output the value received from U and the adversary outputs an arbitrary function of its view. Let us denote the ideal-process output random variables for inputs a, b, ε by $\overline{A}(a), \overline{B}(b), \overline{T}(\varepsilon)$, and $\overline{C}$.

An *actively cheating* A or B is under the complete control of an adversary and may halted at any time. For simplicity, we assume it sends *some* message whenever it is supposed to do so and behaves arbitrarily otherwise.

The active adversary in the *real* process is the same as for the passive case, except that we assume w.l.o.g. that the corrupt party outputs the empty string (its output can always be included in C). In the *ideal* process, the active adversary is similar to the passive case, except that it may cause the corrupt party P to send an arbitrary value i_P to U and that P outputs the empty string.

The security definition combines correctness and privacy by requiring that for any real adversary C, there is an ideal adversary $\overline{C}$ such that for all inputs a, b, ε, the distributions of $(A(a), B(b), T(\varepsilon), C)$ and $(\overline{A}(a), \overline{B}(b), \overline{T}(\varepsilon), \overline{C})$ are computationally indistinguishable (i.e., the distinguisher has access to a and b).

Definition 1. A two-party bidding protocol with an oblivious third party is *secure* if for every passive real adversary C, there is a passive ideal adversary $\overline{C}$ whose running time is bounded by a polynomial in the running time of C such that for all $a \in \mathcal{A}$ and $b \in \mathcal{B}$, the outputs $(A(a), B(b), T(\varepsilon), C)$ and $(\overline{A}(a), \overline{B}(b), \overline{T}(\varepsilon), \overline{C})$ are computationally indistinguishable.

A protocol is *robust* if the same condition holds with C and $\overline{C}$ being active adversaries.

In particular, A should not gain more information about B's input b than what follows from A's input a (or from whatever A could have sent to T) and $f(a, b, \varepsilon)$; likewise, T, should learn nothing about a or b.

3.2 Semi-Honest Case

We first describe the protocol for the semi-honest model, where all participants are supposed to follow the prescribed protocol, but keep all internal records. The extension to a robust protocol that prevents malicious behavior is presented in the next section.

Let $\mathcal{S}$ be a homomorphic public-key cryptosystem as described above such that its message space $\mathcal{X}$ satisfies $|\mathcal{X}| > 2^{k'}$. We need a map $\lambda : \{0,1\}^{k'} \to \mathbb{Z}$ that associates a k'-bit prime p with each k'-bit string x in a deterministic and efficiently invertible way. Denote by $\lambda^{-1}(\cdot)$ the function that on input p returns a random element of $\{x' \in \mathcal{X} | \lambda(x') = p\}$. (For example, interpret x as natural number and use a primality test to find the smallest prime greater than x.)

Setup. T generates a public-key/secret-key pair E_k, D_k and publishes E_k (i.e., sends it to A and B over the secure channels).

A chooses random and independent values $x_l, x_{l-1}, \ldots, x_0 \in \mathcal{X}$ and $s_{l-1}, \ldots, s_0 \in \mathcal{X}$ (in the domain of E_k) and a key κ for the Δ-universal one-way hash function H_k and sends them to B over the secure channel.

Bid preparation. A chooses a random $t_A \in \{0,1\}^{k'}$ that defines a k'-bit prime $p_A = \lambda(t_A)$. Next, she generates a random $m_A \in \mathcal{R}_{k''}$ that hides p_A. For $j = l-1, \ldots, 0$, she computes

$$\delta_{A,j} \;=\; H_k(\kappa, x_j + s_j) \oplus t_A,$$

she chooses $u_j \in \{0,1\}^k$ at random and encrypts

$$y_{A,j} \;=\; \begin{cases} E_k(u_j, x_j - x_{j+1}) & \text{if } a_j = 0 \\ E_k(u_j, x_j - x_{j+1} + s_j) & \text{if } a_j = 1. \end{cases}$$

Similarly, B chooses a random $t_B \in \{0,1\}^{k'}$ that defines a k'-bit prime $p_B = \lambda(t_B)$ and generates a random $m_B \in \mathcal{R}_{k''}$ that hides p_B. For $j = l-1, \ldots, 0$, B computes

$$\delta_{B,j} \;=\; H_k(\kappa, x_j - s_j) \oplus t_B,$$

chooses $u'_j \in \{0,1\}^k$ at random and lets

$$y_{B,j} \;=\; \begin{cases} E_k(u'_j, 0) & \text{if } b_j = 0 \\ E_k(u'_j, -s_j) & \text{if } b_j = 1. \end{cases}$$

A sends $y_{A,l-1}, \ldots, y_{A,0}, \delta_{A,l-1}, \ldots, \delta_{A,0}, m_A$ to B over the secure channel. Then B computes (in $\mathcal{C}$)

$$z_j \;=\; y_{A,j} \cdot y_{B,j}$$

for $j = l-1, \ldots, 0$. He sends

$$\kappa,\ x_l,\ z_{l-1}, \ldots, z_0,\ \delta_{A,l-1}, \ldots, \delta_{A,0},\ \delta_{B,l-1}, \ldots, \delta_{B,0},\ m_A, m_B \tag{1}$$

to T.

Evaluation. T lets $c_l = x_l$. T chooses $g_{A,l} \in \mathrm{QR}_{m_A}$ by selecting a random element of $\mathbb{Z}_{m_A}$ and squaring it and chooses $g_{B,l} \in \mathrm{QR}_{m_B}$ similarly. Then T repeats the following steps for $j = l-1, \ldots, 0$:

1. $c_j = c_{j+1} + D_k(z_j)$;
2. $q_{A,j} = \lambda(H_k(\kappa, c_j) \oplus \delta_{A,j})$; $\quad g_{A,j} = (g_{A,j+1})^{q_{A,j}} \mod m_A$;
3. $q_{B,j} = \lambda(H_k(\kappa, c_j) \oplus \delta_{B,j})$; $\quad g_{B,j} = (g_{B,j+1})^{q_{B,j}} \mod m_B$;

Finally, T chooses $r_A \in \mathbb{Z}_{m_A}$ and $r_B \in \mathbb{Z}_{m_B}$ randomly and independently, computes

$$\begin{aligned} h_A &= (g_{A,0})^{r_A}; \\ h_B &= (g_{B,0})^{r_B}; \end{aligned}$$

and sends h_A over the secure channel to A and h_B over the secure channel to B.

Result extraction. A tests whether h_A has a p_Ath root modulo m_A and outputs 1 (concluding that $a > b$) if the test succeeds, and 0 otherwise. Note that she can test this using the factorization of m_A by checking whether

$$h_A{}^{\phi(m_A)/p_A} \equiv 1 \pmod{m_A}.$$

Similarly, B outputs 1 (concluding $a < b$) if h_B has a p_Bth root and 0 otherwise.

The communication requirements of the protocol are minimal: after T has initially distributed E_k, at most one round of interaction is required between every pair of participants. The protocol can be reordered such that there are only three communication steps.

1. A sends $\kappa; x_l, x_{l-1}, \ldots, x_0;\ s_{l-1}, \ldots, s_0;\ y_{A,l-1}, \ldots, y_{A,0};\ \delta_{A,l-1}, \ldots, \delta_{A,0};\ m_A$ to B.
2. B sends $\kappa; x_l;\ z_{l-1}, \ldots, z_0;\ \delta_{A,l-1}, \ldots, \delta_{A,0};\ \delta_{B,l-1}, \ldots, \delta_{B,0};\ m_A, m_B$ to T.
3. T sends h_A to A and h_B to B.

The idea behind the protocol is that T blindly compares the bits of a and b, starting with the most significant bit. It helps to consider the following relation that can be readily verified: for $j = l-1, \ldots, 0$, we have

$$c_j = x_j + \sum_{j'=j}^{l-1} s_{j'}(a_{j'} - b_{j'}). \tag{2}$$

A and B obtain the result of the comparison using the mechanism of the ΦHA-based PIR protocol.

Theorem 2. *Under the ΦHA and the security assumption for the public-key cryptosystem $\mathcal{S}$, the protocol above is a secure two-party bidding protocol with an oblivious third party.*

Proof (Sketch). According to Definition 1, we have to show that cheating parties do not gain an advantage in the real protocol compared to the ideal process. Because we consider first only passive cheating, the proof that $(a, b, A(a), B(b), T(\varepsilon)) \overset{c}{\approx} (a, b, \overline{A}(a), \overline{B}(b), \overline{T}(\varepsilon))$ establishes the correctness of the protocol and $C \overset{c}{\approx} \overline{C}$ establishes privacy.

Correctness. Fix inputs a and b and assume $a \geq b$. Let j^* be the index of the most significant bit where a and b differ ($a_{j^*} = 1$ and $b_{j^*} = 0$) and let $j^* = -1$ if $a = b$. For $j = l, \ldots, j^* + 1$, we have

$$c_j = x_j \tag{3}$$

and therefore, $q_{A,j}, q_{B,j}$ are essentially random primes. At index j^*, T obtains $c_{j^*} = x_{j^*} + s_{j^*}$ and $q_{A,j^*} = p_A$, but q_{B,j^*} is a random prime. For smaller j, the primes $q_{A,j}, q_{B,j}$ are again essentially random because for $j = j^* - 1, \ldots, 0$,

$$c_j = s_{j^*} + x_j + \sum_{j'=j}^{j^*-1} s_{j'}(a_{j'} - b_{j'}) \tag{4}$$

where s_{j^*} is a random element of $\mathcal{X}$ and the sum consists of random values that are independent of s_{j^*}. The outputs of A and B directly depend on whether h_A and h_B have a p_Ath or p_Bth root, respectively. h_A has a p_Ath root because $q_{A,j^*} = p_A$ and further exponentiating cannot eliminate that.

On the other hand, if $g_{B,l}$ and r_B have no p_Bth root and $q_{B,j} \neq p_B$ for all j, then h_B has no p_Bth root. It is easy to see that both conditions are satisfied except with exponentially small probability (in k') because p_B and $q_{B,j}$ are all random primes of size $\Theta(2^{k'})$ and r_B is random.

The other cases $a < b$ and $a = b$ follow analogously. This shows that the outputs $A(a)$, $B(b)$, and $T(\varepsilon)$ agree with $f(a, b, \varepsilon)$ except with negligible probability.

Privacy. The proof is by contradiction. Suppose the protocol is not secure. Then there is a probabilistic polynomial-time real adversary $\widetilde{C}$ such that *no* corresponding polynomial-time ideal adversary $\overline{C}$ exists that achieves

$$(a, b, A(a), B(b), T(\varepsilon), \widetilde{C}) \overset{c}{\approx} (a, b, \overline{A}(a), \overline{B}(b), \overline{T}(\varepsilon), \overline{C})$$

for all a, b.

Cheating A or B. Suppose $\widetilde{C}$ has access to B's communication and internal records. What such a $\widetilde{C}$ might do, for which there is no corresponding $\overline{C}$, is to gain more information about A's input a than what follows from B's own output of $f(a, b, \varepsilon)$. In other words, w.l.o.g. there are values $a', a'', b \in [0, 2^l - 1]$ with $a' \neq a''$ and $a' > b$ and $a'' > b$ such that $\widetilde{C}$ can distinguish A's input a' from a'' by observing B. Let C' denote $\widetilde{C}$'s output random variable if A's input is a' and B's input is b and let C'' denote $\widetilde{C}$'s output random variable if A's input is a'' and B's input is b. C' and C'' are distinguishable by a probabilistic polynomial-time algorithm with noticeable probability.

B's (and $\widetilde{C}$'s) view consists of κ, $x_l, \ldots, x_0$, $s_{l-1}, \ldots, s_0$, $\delta_{A,l-1}, \ldots, \delta_{A,0}$, $\delta_{B,l-1}, \ldots, \delta_{B,0}$, $y_{A,l-1}, \ldots, y_{A,0}$, $y_{B,l-1}, \ldots, y_{B,0}$, m_A, m_B, and h_B; suppose for the moment that it also includes p_A and the integer factorization of m_A. The only part that the adversary cannot simulate itself is $y_{A,l-1}, \ldots, y_{A,0}$ and h_B.

As $a' > b$ and $a'' > b$, the adversary cannot obtain its advantage from T's response h_B: in both cases, T starts with $g_{B,l}$ that has order $\phi(m)/2$ except with exponentially small probability, and T raises this modulo m_B to primes $q_{B,j}$ and to a random $r_B \in \mathbb{Z}_{m_B}$ that are relatively prime to $\phi(m)/2$ except with exponentially small probability. According to Lemma 1, T's replies to B are statistically indistinguishable and this cannot cause a noticeable difference between C' and C''.

This implies that $\widetilde{C}$ gains its advantage only from observing $y_{A,l-1}, \ldots, y_{A,0}$ and $\delta_{A,l-1}$, $\ldots$,$\delta_{A,0}$. We can use $\widetilde{C}$ to construct an algorithm $\widetilde{D}$ to break the security assumption of $\mathcal{S}$ as follows. Let $\mathcal{J} \subseteq [0, l-1]$ be the indices of the bits where a' and a'' differ.

$\widetilde{D}$ takes as inputs $a \in [0, 2^l - 1]$ and $w \in \mathcal{X}$. $\widetilde{D}$ chooses a random $j^* \in \mathcal{J}$ and emulates the bidding protocol for A, B, and T with inputs a, b, ε. In the emulation of A, $\widetilde{D}$ replaces x_{j^*} by $w + x_{j^*+1}$ if $a_{j^*} = 0$ or by $w + x_{j^*+1} - s_{j^*}$ if $a_{j^*} = 1$. The emulation of the protocol is continued and the adversary $\widetilde{C}$ is called with the view of B as input. $\widetilde{D}$ outputs whatever $\widetilde{C}$ outputs.

Above we have defined how $\widetilde{D}$ operates on arbitrary inputs. If $\widetilde{D}$ is run on inputs a' and a randomly chosen $w' \in \mathcal{X}$, then its output has the same distribution as C' by construction. If $\widetilde{D}$ is run on a'' and an independently chosen random $w'' \in \mathcal{X}$, the output distribution is C''.

Because C' and C'' are distinguishable, $\widetilde{D}$ can be used to distinguish encryptions of $w' \neq w''$ with noticeable probability, contradicting the semantic security of $\mathcal{S}$.

A passively cheating A can be handled analogously.

Cheating T. For the second part of the proof, suppose the adversary $\widetilde{C}$ has access to T's view, which includes the values in (1) and their decryptions. What such a $\widetilde{C}$ might do, for

which there is no corresponding $\overline{C}$, is to gain information about A's or B's input a and b. W.l.o.g. there are values $a', a'', b', b'' \in [0, 2^l - 1]$ with $a' \neq a''$ or $b' \neq b''$ such that $\widetilde{C}$ can with noticeable probability distinguish A and B's inputs a' and b' from a'' and b'' by observing T's view. Assume w.l.o.g. $a' > a''$ and $b' = b''$.

Define C' as the random variable of $\widetilde{C}$'s output for A and B's inputs a' and b' and define C'' analogously for a'' and b''. There is a probabilistic polynomial-time algorithm that can distinguish C' and C'' with noticeable probability.

From T's view, $\widetilde{C}$ can gain information about A's inputs either through a relation among the x_j, s_j or by exploiting $p_A|\phi(m_A)$. We show that the first case implies collisions in $\mathcal{H}$ and the second case contradicts the ΦHA.

Suppose $\widetilde{C}$ exploits the system of $2l$ equations

$$\begin{aligned} t_A &= H_k(\kappa, x_j + s_j) \oplus \delta_{A,j} \\ t_B &= H_k(\kappa, x_j - s_j) \oplus \delta_{B,j} \end{aligned}$$

for $j = 0, \ldots, l-1$ with unknown t_A, t_B. As shown in the correctness part, T knows only the values in (3) and (4) and they satisfy at most one equation in the system above. If $\widetilde{C}$ is able to determine the value side of a second equation, this implies $H_k(\kappa, x_j + s_j) = H_k(\kappa, x') \oplus \delta'$ for some j, some $x' \neq x_j + s_j$ and some δ' that is a sum of δ_A's or δ_B's. But this corresponds to a collision in the Δ-universal one-way hash function $\mathcal{H}$.

In the other case, we use $\widetilde{C}$ to construct an algorithm $\widetilde{D}$ to contradict the ΦHA as follows. $\widetilde{D}$ takes as inputs two numbers $p \in \mathcal{P}_{k'}$, $m \in \mathcal{R}_{k''}$ and two values $a, b \in [0, 2^l - 1]$. $\widetilde{D}$ emulates the protocol for $A(a)$ and $B(b)$ as prescribed, except that it modifies A to choose $t_A = \lambda^{-1}(p)$ and $m_A = m$. Then $\widetilde{C}$ is called with T's view as input and $\widetilde{D}$ outputs whatever $\widetilde{C}$ outputs.

Observe that for a random $m \in \mathcal{R}_{k''}$ and $p_0 \in \mathcal{P}_{k'}$ that is hidden by m, the output distribution of $\widetilde{D}(m, p_0, a', b')$ is equal to C' by construction (and analogously, $\widetilde{D}(m, p_0, a'', b'')$ is equal to C''). Define D' as the output distribution of $\widetilde{D}(m, p_1, a', b')$ for a random $m \in \mathcal{R}_{k''}$ and an independently chosen random $p_1 \in \mathcal{P}_{k'}$. Then D' is efficiently distinguishable from either C' or C'' with noticeable probability and this contradicts the ΦHA. □

3.3 Robust Protocol

The above protocol is not robust: an actively cheating A or B might gain information on the other party's input or disrupt the joint computation of $f(a, b, \varepsilon)$. For example, A might choose the x_j and s_j such that T maps them to some prime $\widetilde{p} \neq p_A$ that also divides m_A and retrieve information about b in this way. As T is cheating only passively, we use it to ensure fairness between A and B, meaning that A learns its component of the result $f(a, b, \varepsilon)$ if and only if B learns it.

The basic idea is to have A and B check each other for compliance with the protocol specification. In particular, A and B prove to each other that they have constructed their inputs to T correctly. A and B both send the same inputs to T, who checks that they are equal. In case of mismatch, T aborts and outputs a special symbol; otherwise, it proceeds as above. Furthermore, we can no longer have A choose the x_j and s_j. A joint random selection protocol based on a commitment scheme is used instead.

The modifications to the protocol in the previous section are as follows.

Setup. A generates $2l + 1$ uniformly random values $\sigma_0, \ldots, \sigma_{2l} \in \mathcal{X}$ and commits to them, obtaining $\gamma_A = Com_k(\rho_A, \sigma_0 \| \cdots \| \sigma_{2l})$. A sends γ_A to B. B chooses $2l + 1$ uniformly random values $\tau_0, \ldots, \tau_{2l} \in \mathcal{X}$, commits to them in $\gamma_B = Com_k(\rho_B, \tau_0 \| \cdots \| \tau_{2l})$, and sends γ_B to A.

Only after receiving the other party's commitment does A reveal $\rho_A, \sigma_0, \ldots, \sigma_{2l}$ and B reveal $\rho_B, \tau_0, \ldots, \tau_{2l}$ (over the secure channel).

A checks that $Ver_k(\gamma_B, \rho_B, \tau_0 \| \cdots \| \tau_{2l}) = 1$. If not, A sends a special symbol abort to T, outputs abort, and halts. B checks that $Ver_k(\gamma_B, \rho_B, \sigma_0 \| \cdots \| \sigma_{2l}) = 1$. If not, B sends a special symbol abort to T, outputs abort, and halts.

Otherwise, both A and B obtain $x_j = \sigma_j + \tau_j$ for $j = l, l-1, \ldots, 0$ and $s_j = \sigma_{j+l} + \tau_{j+l}$ for $j = 1, \ldots, l$.

Bid preparation. After A has determined p_A, m_A and the $\delta_{A,j}$ as above, she forwards p_A, m_A, $\delta_{A,l-1}, \ldots, \delta_{A,0}$, and the integer factorization of m_A to B.

B computes p_B, m_B and the $\delta_{B,j}$ as above. He sends p_B, m_B, $\delta_{B,l-1}, \ldots, \delta_{B,0}$ to A, together with the integer factorization of m_B.

A checks that $m_B \in \mathcal{R}_{k''}$ (using an efficient primality test) and that $p_B | \phi(m_B)$. A also tests that

$$p_B \;=\; \lambda(H_k(\kappa, x_j - s_j) \oplus \delta_{B,j})$$

for $j = l-1, \ldots, 0$. If a test fails, A sends a special symbol abort to T, outputs abort, and halts. Otherwise, A computes the encryptions $y_{A,j}$ as above and sends

$$\kappa,\, x_l,\, z_{l-1}, \ldots, z_0,\, \delta_{A,l-1}, \ldots, \delta_{A,0},\, \delta_{B,l-1}, \ldots, \delta_{B,0},\, m_A, m_B \tag{5}$$

to T over the secure channel.

Similarly, B checks that $m_A \in \mathcal{R}_{k''}$ and that $p_A | \phi(m_A)$. B also tests that

$$p_A \;=\; \lambda(H_k(\kappa, x_j + s_j) \oplus \delta_{A,j})$$

for $j = l-1, \ldots, 0$. If any test fails, B sends abort to T, outputs abort, and halts. Otherwise, B computes the encryptions $y_{B,j}$ as above and sends

$$\kappa,\, x_l,\, z_{l-1}, \ldots, z_0,\, \delta_{A,l-1}, \ldots, \delta_{A,0},\, \delta_{B,l-1}, \ldots, \delta_{B,0},\, m_A, m_B \tag{6}$$

to T over the secure channel.

Evaluation. T compares the values received from A and from B. In case one of the messages was abort or if some values in (5) and in (6) are not equal, T sends back abort to A and B, outputs abort itself, and halts. Otherwise, T proceeds as above.

Result extraction. If A (or B) receive abort, output abort and halt; otherwise same as above.

Because neither party can influence the random choice of the x_j and s_j, they correspond to the random choices in the semi-honest protocol (except with negligible probability). It is easy to see that active cheating by A (or B) can succeed in producing $m_A \notin \mathcal{R}_{k''}$ (or m_B) only with negligible probability. It follows from Lemma 1 that h_A (and h_B) do not reveal more information than in the protocol for the semi-honest case.

Because T answers only if both parties agree on their submitted values, deviating from the protocol is prevented (input substitution by the ideal adversary $\overline{C}$ may not occur). This eliminates all attacks that are not also possible in the passive case and reduces the security proof to Theorem 2.

Theorem 3. *Under the ΦHA and the security assumption for the public-key cryptosystem $\mathcal{S}$, the modified protocol above is a robust two-party bidding protocol with an oblivious third party.*

4 Auctions with Two Servers

Based on the private bidding primitive, we can realize a protocol for sealed-bid auctions among n parties $P_1, \ldots, P_n$ with secret bids $a_1, \ldots, a_n$ and two semi-trusted auction servers T and V. T plays the same role as in the two-party protocol, obliviously comparing two bids. V chooses the random values for n instances of the private bidding protocol. The bidders encrypt their bids, send them to V, but are not involved further. V determines the highest bid through n successive queries to T and learns a partial order of $a_1, \ldots, a_n$, but not more.

A more detailed description follows (without proof). The participants are $P_1, \ldots, P_n$, T and V. The input to P_r is an l-bit number $a_r \in [0, 2^l - 1]$; T and V have no inputs. The only output of the protocol is a number $r_{\max} \in [1, n]$ produced by V such that $r_{\max} = \arg\max_{r \in [1,n]} a_r$.

We assume $P_1, \ldots, P_n$ and T are connected to V by secure channels.

During setup, T generates a public-key/secret-key pair E_k, D_k and publishes E_k. V chooses random values for $i \in [1, n]$

- $\kappa_i \in \mathcal{K}$ and $x_{i,l}, x_{i,j}, s_{i,j} \in \mathcal{X}$ for $j \in [0, l-1]$,
- $t_i^A, t_i^B \in \{0,1\}^{k'}$,
- $p_i^A \in \mathcal{P}_{k'}$ and $m_i^A \in \mathcal{R}_{k''}$ hiding p_i^A and $p_i^B \in \mathcal{P}_{k'}$ and $m_i^B \in \mathcal{R}_{k''}$ hiding p_i^B,

to be used in n parallel copies of the two-party private bidding protocol. V determines $\delta_{i,j}^A = H_k(\kappa_i, x_{i,j} + s_{i,j}) \oplus t_i$ and $\delta_{i,j}^B = H_k(\kappa_i, x_{i,j} - s_{i,j}) \oplus t_i$. V sends all values to $P_1, \ldots P_n$ over the secure channels.

Each party P_r prepares two encryptions of its bid a_i per parallel copy as in the standard protocol, one in A's role and one in B's role. Two copies are needed because it is yet unknown whether P_r will play the role of A or B. The values returned from P_r to V are

$$y_{r,i,j}^A = \begin{cases} E_k(u_{r,i,j}, x_{i,j} - x_{i,j+1}) & \text{if } a_{r,j} = 0 \\ E_k(u_{r,i,j}, x_{i,j} - x_{i,j+1} + s_{i,j}) & \text{if } a_{r,j} = 1 \end{cases}$$

and

$$y_{r,i,j}^B = \begin{cases} E_k(u'_{r,i,j}, 0) & \text{if } a_{r,j} = 0 \\ E_k(u'_{r,i,j}, -s_{i,j}) & \text{if } a_{r,j} = 1 \end{cases}$$

for $i \in [1, n]$ and $j \in [0, l-1]$.

V chooses a random permutation $\pi_1, \ldots, \pi_n$ of $[1, n]$ and lets $i_{\max} = 1$. To find the winning bidder, V and T repeat for $i = 2, \ldots, n$:

1. V compares the bids of P_{π_i} and $P_{\pi_{i_{\max}}}$ using T's help; V computes $z_{i,j} = y_{\pi_i,i,j}^A \cdot y_{\pi_{i_{\max}},i,j}^B$ and sends $x_{i,l}$, $z_{i,j}$, $\delta_{i,j}^A$, $\delta_{i,j}^B$, m_i^A, m_i^B to T for $j \in [0, l-1]$.
2. T performs the evaluation step on the received values as in the two-party protocol and sends back the numbers $h_i^A \in \mathbb{Z}_{m_i^A}$ and $h_i^B \in \mathbb{Z}_{m_i^B}$.
3. V determines the winner (P_{π_i} or $P_{\pi_{i_{\max}}}$) by testing whether h_i^A has a p_i^Ath root. If and only if the test succeeds does V set $i_{\max}$ to i.

Finally, V outputs $r_{\max} = \pi_{i_{\max}}$, thereby declaring that party $P_{r_{\max}}$ wins the auction.

It is straightforward to verify that the protocol is correct and private, with the exception that V learns the partial order of the bids, but nothing else about them. A semi-honest T does not gain information about the bids; but if a malicious T conspires with at least one bidder, it could see all comparisons in the open, so that all servers can see the bids.

The bidders need only one round of interaction with the auction service; the auction servers need $O(n)$ rounds of interaction. The workload (and communication complexity) of one bidder is $O(kln)$, whereas the total work of the auction is $O(kln^2)$.

References

[AF90] Martin Abadi and Joan Feigenbaum, *Secure circuit evaluation: A protocol based on hiding information from an oracle*, Journal of Cryptology **2** (1990), 1–12.

[ASW97] N. Asokan, Matthias Schunter, and Michael Waidner, *Optimistic protocols for fair exchange*, Proc. 4th ACM Conference on Computer and Communications Security, 1997, pp. 6, 8–17.

[ASW98] N. Asokan, Victor Shoup, and Michael Waidner, *Optimistic fair exchange of digital signatures*, Advances in Cryptology: EUROCRYPT '98 (Kaisa Nyberg, ed.), Lecture Notes in Computer Science, vol. 1403, Springer, 1998.

[Bea91] Donald Beaver, *Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority*, Journal of Cryptology **4** (1991), no. 2, 75–122.

[Ben94] Josh Benaloh, *Dense probabilistic encryption*, Proc. Workshop on Selected Areas of Cryptography (Kingston, ON), May 1994, pp. 120–128.

[BGMR90] Michael Ben-Or, Oded Goldreich, Silvio Micali, and Ronald L. Rivest, *A fair protocol for signing contracts*, IEEE Transactions on Information Theory **36** (1990), no. 1, 40–46.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson, *Completeness theorems for non-cryptographic fault-tolerant distributed computation*, Proc. 20th Annual ACM Symposium on Theory of Computing (STOC), 1988, pp. 1–10.

[BS98] Carrie Beam and Arie Segev, *Auctions on the internet: A field study*, Working Paper 98-WP-1032, Fisher Center for Management and Information Technology, Haas School of Business, University of California, Berkeley, 1998.

[Can98] Ran Canetti, *Security and composition of multi-party cryptographic protocols*, Report 98-18, Theory of Cryptography Library, 1998.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård, *Multiparty unconditionally secure protocols*, Proc. 20th Annual ACM Symposium on Theory of Computing (STOC), 1988, pp. 11–19.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler, *Computationally private information retrieval with polylogarithmic communication*, Advances in Cryptology: EUROCRYPT '99 (Jacques Stern, ed.), Lecture Notes in Computer Science, vol. 1592, Springer, 1999.

[FR96] Matthew K. Franklin and Michael K. Reiter, *The design and implementation of a secure auction service*, IEEE Transactions on Software Engineering **22** (1996), no. 5, 302–312.

[FR97] Matthew K. Franklin and Michael K. Reiter, *Fair exchange with a semi-trusted third party*, Proc. 4th ACM Conference on Computer and Communications Security, 1997.

[GM84] Shafi Goldwasser and Silvio Micali, *Probabilistic encryption*, Journal of Computer and System Sciences **28** (1984), 270–299.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson, *How to play any mental game or a completeness theorem for protocols with honest majority*, Proc. 19th Annual ACM Symposium on Theory of Computing (STOC), 1987, pp. 218–229.

[Gol98] Oded Goldreich, *Secure multi-party computation*, Manuscript, 1998, (Version 1.1).

[HTK98] Michael Harkavy, Doug Tygar, and Hiroaki Kikuchi, *Electronic auctions with private bids*, Proc. 3rd USENIX Workshop on Electronic Commerce (Boston), 1998.

[KF98] Manoj Kumar and Stuart I. Feldman, *Internet auctions*, Proc. 3rd USENIX Workshop on Electronic Commerce (Boston), 1998.

[Mic97] Silvio Micali, *Certified e-mail with invisible post offices*, Invited presentation at the RSA '97 conference, 1997.

[MR92] Silvio Micali and Phillip Rogaway, *Secure computation*, Advances in Cryptology: CRYPTO '91 (Joan Feigenbaum, ed.), Lecture Notes in Computer Science, vol. 576, Springer, 1992, pp. 392–404.

[NS98] David Naccache and Jacques Stern, *A new public-key cryptosystem based on higher residues*, Proc. 5th ACM Conference on Computer and Communications Security, 1998, pp. 59–66.

[NY89] Moni Naor and Moti Yung, *Universal one-way hash functions and their cryptographic applications*, Proc. 21st Annual ACM Symposium on Theory of Computing (STOC), 1989, pp. 33–43.

[Yao82] Andrew C.-C. Yao, *Protocols for secure computation*, Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS), 1982, pp. 160–164.

A Homomorphic Encryption Schemes

We describe two ways to implement the homomorphic public-key cryptosystem $\mathcal{S}$ with plaintext space $\mathcal{X}$ in the sense of Section 2.4. Given a homomorphic public-key system $\mathcal{S}'$ with plaintext space $\mathcal{X}'$ of cardinality $M \geq 3$, such a system can be built using $\lceil \log_M |\mathcal{X}| \rceil$-fold application of $\mathcal{S}'$ and a suitable M-ary representation of $\mathcal{X}$.

A.1 The Benaloh System

Benaloh's *dense probabilistic encryption* [Ben94] is based on the higher-residuosity assumption; its underlying principle is closely related to the ΦHA. For some odd $r = O(\log k)$, a key is generated as follows. Choose $n = pq$ as the product of two k-bit primes p and q such that $r|\phi(n) = (p-1)(q-1)$, but r and $(p-1)(q-1)/r$ are relatively prime. Choose some $y \in \mathbb{Z}_n^*$ such that $y^{(p-1)(q-1)/r} \not\equiv 1 \pmod{n}$. The public key is the pair (n, y) and the secret key is (p, q).

The encryption function $E_{k,r} : \mathbb{Z}_n^* \times [0, r-1] \to \mathbb{Z}_n^*$ is given by $E_{k,r}(u, x) = y^x u^r \mod n$ for a randomly chosen $u \in \mathbb{Z}_n^*$.

Decryption is done by exploiting that a ciphertext $z \in \mathbb{Z}_m^*$ encrypts plaintext 0 if and only if $z^{(p-1)(q-1)/r} \equiv 1 \pmod{n}$ (i.e., if z has rth roots modulo n). Because r is small, decryption is done by exhaustive search for the smallest $x \in [0, r-1]$ such that $y^{-x}z$ is an encryption of 0. The decryption complexity can be lowered to $O(\sqrt{r})$ by using the baby-step giant-step method.

The semantic security of this system is equivalent to the *higher-residuosity assumption*: for given r, a randomly chosen public key (n, y), a random $z_0 \in \mathbb{Z}_n^*$ that has rth roots and a random $z_1 \in \mathbb{Z}_n^*$ with no rth roots, (n, y, z_0) and (n, y, z_1) are computationally indistinguishable.

A.2 The Naccache-Stern System

This system proposed by Naccache and Stern [NS98] works by embedding a trap door in the discrete logarithm and hiding it by an RSA modulus of unknown factorization. It is much more efficient than the Benaloh system in terms of the expansion rate (which can even be linear according to the authors), but it needs a stronger security assumption.

For some small $B = O(\log k)$, let r be a square-free odd B-smooth integer of length $O(k')$ bits; the rest is almost the same as the Benaloh system. Let $n = pq$ be a product of two k-bit primes p and q such that $r|\phi(n) = (p-1)(q-1)$, but r and $(p-1)(q-1)/r$ are relatively prime. Choose some $g \in \mathbb{Z}_n^*$ the order of which is a large multiple of r in $\mathbb{Z}_n^*$. Methods to generate such n are discussed in [NS98]. A public key is a triple (n, r, g), and the corresponding secret key is (p, q).

The encryption function $E_k : \mathbb{Z}_n^* \times [0, r-1] \to \mathbb{Z}_n^*$ is given by $E_k(u, x) = g^x u^r \mod n$ for a randomly chosen $u \in \mathbb{Z}_n^*$.

The difference to the Benaloh system lies in the decryption method because exhaustive search over $x \in [0, r-1]$ would take exponential time. Decryption is done by exploiting the smoothness of r to compute the discrete logarithm to base g, using Chinese remaindering. Details can be found in [NS98].

The semantic security of this system is equivalent to the higher-residuosity assumption in the case where n is of the special form described above, such that $\phi(n)$ contains a B-smooth divisor r.