A Two-Server, Sealed-Bid Auction Protocol (Extended Abstract)

Ari Juels and Michael Szydlo

RSA Laboratories
Bedford, MA 01730, USA
E-mail: {ajuels,mszydlo}@rsasecurity.com

Abstract. Naor, Pinkas, and Sumner introduced and implemented a sealed-bid, two-server auction system that is perhaps the most efficient and practical to date. Based on a cryptographic primitive known as oblivious transfer, their system aims to ensure privacy and correctness provided that at least one auction server behaves honestly. As observed in [19], however, the NPS system suffers from a security flaw in which one of the two servers can cheat so as to modify bids almost arbitrarily and without detection. We propose a means of repairing this flaw while preserving the attractive practical elements of the NPS protocol, including minimal round complexity for servers and minimal computation by players providing private inputs. Our proposal requires a slightly greater amount of computation and communication on the part of the two auction servers, but actually involves much *less* computation on the part of bidders. This latter feature makes our proposal particularly attractive for use with low-power devices. While the original proposal of NPS involved several dozen exponentiations for a typical auction, ours by contrast involves only several dozen modular multiplications.
The key idea in our proposal is a form of oblivious transfer that we refer to as *verifiable proxy oblivious transfer* (VPOT). The security of VPOT is predicated in a provable manner on a collection of common cryptographic assumptions, including the RSA assumption, quadratic residuosity assumption, and the random oracle model.

Key words: auction, sealed-bid auction, oblivious transfer, secure multiparty computation, secure function evaluation

1 Introduction

Cryptography offers a broad range of tools for distributing trust among computing entities in flexible and often unexpected ways. In an electronic auction setting, for example, a foundational cryptographic procedure known as *secure function evaluation* enables the submission and processing of sealed bids without the presence of a single, trusted auctioneer. As secure function evaluation is rather impractical in its general form, a large body of research, e.g., [1, 5, 11, 17, 19, 24], has focused on tailoring cryptographic protocols specifically to achieve efficient sealed-bid auction systems.

A recent architecture proposed and implemented by Naor, Pinkas, and Sumner [20] represents substantial progress toward the goal of practical sealed-bid auctioning with distributed trust. In their system, bidders submit encrypted bids to a front-end server known as an *auctioneer*. With the involvement of a second, back-end server known as an *auction issuer*, any type of sealed-bid auction may be conducted, e.g., highest-bid auctions, Vickrey auctions, and so forth. The architecture aims to distribute trust between the two servers in the following simple model: If at least one server is honest, the bids of all participants will remain private, and any auction outcome is assured to be correct. There is no robustness, however, in the sense that either server can cause the protocol to terminate. NPS report good scalability, claiming that their system can accommodate hundreds or even thousands of bidders with reasonable overhead. We note that the computational requirement for bidders in their system is approximately one modular exponentiation per bit in a bid representation. See [20] for further details.

As identified in a footnote in work by Jakobsson and Juels [19], however, the NPS system has a serious flaw that permits tampering by one of the servers. Although not explicated in [19], it is easy to see that the auction issuer can modify any bit in any bid without detection. The underlying problem is a variant introduced by NPS on a cryptographic primitive known as *1-out-of-2 oblivious transfer* (1-2 OT), as we now explain.

Basic 1-2 OT is a procedure involving two players, a *Chooser* and a *Sender*. The Sender possesses a pair of values (t_0, t_1). We refer to these values throughout our paper as *tags*. The Chooser elects to receive from the Sender one of these two tags t_b for $b \in \{0, 1\}$. The 1-2 OT procedure is oblivious in the sense that the Sender learns negligible information about b. An additional privacy property of 1-2 OT that the Chooser learns negligible information about t_{1-b}, i.e., the value that she did not select. NPS consider a variant on 1-2 OT, called *proxy oblivious transfer*. This variant involves an intervening third party known as a *Proxy*, who receives the value t_b on behalf of the Chooser, but herself learns negligible information about b and t_{1-b}. We provide details on the protocol below. While proxy oblivious transfer accomplishes the goal for which it was designed, namely privacy protection, it does not include any mechanism for *verifiability*. In particular, the proxy oblivious transfer protocol does not ensure that the Sender transmitted t_b as desired. In the NPS auction setting, in particular, the Sender (auction issuer) can substitute the tag t_{1-b}. This means that the auction issuer can tamper with bids.

In this paper, we introduce a protocol called *verifiable proxy oblivious transfer* (VPOT) that addresses the vulnerability in the NPS protocol. In principle, introducing verifiability into proxy oblivious transfer is not difficult using basic – and potentially expensive – cryptographic techniques such as zero-knowledge proofs. Our contribution in the design of VPOT is a collection of techniques that render the verification process computationally inexpensive and yet, at the same time, provably secure. When VPOT is introduced into the NPS auction protocol, it increases the computational burden on auction servers somewhat, but actually results in much *less* computation for bidders. This is particularly desirable given the fact that bidders in many settings may wish to employ low-power, handheld devices. Thus, VPOT not only remedies the security flaw in the NPS architecture, but renders the system even more practical.

1.1 Background: Two-party computation and the NPS protocol

Secure function evaluation (also known as secure multi-party computation) began with the work of Yao [25], and Goldreich, Micali, and Wigderson [15], and has spawned an ever growing body of research papers. See [13] for a good overview of early work. The general goal of secure multi-party computation is to enable m players to apply a function F to respective private inputs $X_1, X_2, \ldots, X_m$ such that some subset of players learns $F(X_1, X_2, \ldots, X_m)$, but no player learns additional, non-negligible information. Privacy and robustness against active (indeed, adaptive) adversaries are possible to achieve provided that the adversary controls at most a fraction of the players. Assuming the existence of a broadcast channel, this fraction is 1/2; otherwise, it is 1/3. For some recent work representing state-of-the-art attempts to achieve practical multi-party protocols, see, e.g., [8, 18].

The two-player case of secure function evaluation is distinguished by its relative simplicity and practicality. The original secure function evaluation protocol of Yao [25] treated this case, and remains an important tool even now. In contrast to more general techniques, in which field operations such as addition and multiplication are the atomic unit of computation, the Yao protocol involves direct computation on boolean gates. While this is a limitation in the general case, many real-world protocols such as auctions involve intensive bitwise manipulation such that boolean circuits are in fact a natural form of representation for the required functions. The Yao protocol is appealing for other reasons as well. Provided that only one player is to learn the output, it is in fact possible to execute the Yao protocol with only one-round of interaction, an observation first set forth implicitly in [20] and explored in detail in [7]. While constant-round secure function evaluation is possible for multiple players, it requires both high overhead and the availability of a broadcast channel [3]. A model in which both players in the Yao protocol learn the output of such computation in a fair manner (given a passive trusted entity) is also possible, and is explored in [6].

Suppose that Alice and Bob wish to engage in the Yao protocol on respective private inputs X_A and X_B such that Bob learns the output $y = F(X_A, X_B)$. Alice constructs a "garbled" circuit representing F. She sends this circuit to Bob, along with a "garbled" representation of X_A. In order to evaluate the "garbled" circuit, Bob additionally needs a "garbled" version of his input X_B. He obtains this from Alice using basic 1-2 *oblivious transfer* (OT) [21], or some enhanced variant. This is the component of the Yao protocol that we focus on in detail in this paper. In the case where Alice may cheat, another important component in two-player secure function evaluation protocols are proofs of correct construction of the Yao circuits. A cut-and-choose protocol for this is proposed in [20], while [7] explores use of general non-interactive proof techniques. (If Alice wishes Bob to send y to her in such a way that she can verify its correctness, she need merely embed a verification tag in her "garbled" version of F in the appropriate manner.)

Numerous variants of oblivious transfer have been considered and compared in the literature [13]. Notions of combining bit commitment with oblivious transfer in a theoretical setting to achieve a committed or verifiable oblivious transfer are explored for example in [10] and [9]. These works explore theoretical approaches that treat oblivious transfer and bit commitment as black boxes, and are thus necessarily expensive. An alternative approach proposed in the literature is one using a trusted initializer [22]. The key observation made by NPS in their auction system design is that by involving a Proxy in the oblivious transfer procedure, it is possible to expand application of basic Yao style two-server function evaluation in such a way that inputs may be accepted from an arbitrarily large number of players, i.e., bidders, while the evaluation process is restricted to two auction servers.

Briefly stated, the NPS construction is as follows. The auction issuer (again, the back-end server) constructs a “garbled” representation of a function F that describes the auction protocol. The auctioneer (again, the front-end server) evaluates the circuit for F using “garbled” inputs representing the bids. In order to obtain the “garbled” input for a bit b in a given bid, it is necessary to invoke the proxy oblivious transfer protocol. In this protocol, the bidder plays the role of the Chooser, the auctioneer plays the role of the Proxy, and the auction issuer plays the role of the Sender. The Sender transmits “garbled” inputs (t_0, t_1) for the circuit corresponding to a '0' bit and a '1' bit in the bid. The Chooser selects t_b, which the Proxy receives through the transfer protocol. Having done this for all bits in all bids, the Proxy is able to evaluate F on the input bids and determine the outcome of the auction. The privacy properties of the proxy oblivious transfer protocol ensure that the Proxy does not learn b or t_{1-b} for any bit. The Proxy therefore does not learn any bidding information and can only evaluate F on correct bid amounts. Likewise, the Sender does not learn the bid amounts. Only if the Proxy and Sender collude is this privacy guarantee breached.

NPS include some other basic security enhancements to the protocol. In particular, for the auctioneer to ensure that the auction issuer has constructed F correctly, the two must engage in a cut-in-choose protocol. Thus, the auctioneer must in fact evaluate multiple, independent circuits representing F. We provide more details below.

1.2 Our contribution: Verifiable proxy oblivious transfer (VPOT)

The failing in the NPS protocol is that the auction issuer can transmit t_{1-b} instead of t_b without detection. To address this problem, we propose a protocol known as verifiable proxy oblivious transfer (VPOT). VPOT enables the Proxy (auctioneer) to ensure that the Sender (auction issuer) sent t_b, as required. VPOT otherwise retains all of the privacy characteristics of proxy oblivious transfer.

An simplified overview of VPOT is as follows. The Sender provides commitments $\mathcal{C}_0$ and $\mathcal{C}_1$ to tags t_0 and t_1 (respectively representing a '0' bit and '1' bit in a bid). These commitments take the form of a randomly ordered pair $(\mathcal{C}_a, \mathcal{C}_{1-a})$, i.e., a is a randomly selected bit. The Sender also provides a commitment $E[a]$ to the ordering a. Observe that the triple $(\mathcal{C}_0, \mathcal{C}_1, E[a])$ binds the Sender to values for t_0 and t_1.

As usual in a 1-2 OT protocol, the Chooser selects a value t_b to be decommitted by the Sender. The Chooser in fact splits this bit b into two shares b_P and b_S such that $b = b_P \oplus b_S$. The Chooser sends the share b_S to the Sender. This is transmitted (via the Proxy) as a ciphertext $E[b_S]$. She sends the share b_P to the Proxy, also in a specially committed form that we do not describe here. It is the splitting of b into two shares that ensures privacy with respect to the two auction servers (provided that there is no collusion).

Finally, the Chooser transmits to the Proxy a secret value x that enables the Proxy to receive the selected tag t_b. The Sender decommits t_b for the Proxy, who then checks the correctness of the decommitment.

In addition to careful protocol decomposition and ordering of steps, here is a list of the more interesting cryptographic techniques used in the construction of

VPOT. While none is individually novel *per se*, our new constructions combine them in a novel way, effectively providing a new fundamental building block useful for securely extending traditional two-party techniques to a setting with multiple contributors.

- *Double commitment:* The Sender's commitment $\mathcal{C}_k(t)$ on tag t in fact consists of a pair of values (Y_1, Y_2). The first value, Y_1, is the commitment on a key or witness k. In particular here, $Y_1 = H(k^3)$, where the cubing operation takes place over an RSA modulus provided by the Sender (as discussed in more detail below). H here is a hash function (modelled as a random oracle for security proofs on the system). Observe that as the hash of a cube, Y_1 is really a commitment within a commitment. It is for this reason that we refer to $\mathcal{C}_k(t)$ as a *double* commitment. The second value of the commitment pair, Y_2, represents an encryption of t under k. In particular, $Y_2 = H(k) \oplus t$, where $\oplus$ denotes the bitwise XOR operator. Observe that knowledge of the witness k is sufficient both to open the commitment and obtain t and also to verify that the commitment has been correctly opened. This double commitment scheme may be seen to be both computationally binding and computationally hiding under the RSA assumption, with the random oracle model invoked for H.
- *RSA-based oblivious transfer:* Most oblivious transfer protocols designed for practical use in two-party secure function evaluation, e.g., in [20, 2], employ El Gamal-based encryption of tags [14]. The result is that the Chooser must perform at least one exponentiation per 1-2 OT invocation. In contrast, we introduce an RSA-based 1-2 OT scheme as the foundation for VPOT. The result is that the Chooser need only perform one RSA cubing, i.e., two modular multiplications, per 1-2 OT invocation. When employed in VPOT, this idea reduces the work of the Chooser by over an order of magnitude with respect to the proxy oblivious transfer protocol of NPS.
- *Goldwasser-Micali encryption:* The encryption function E in our brief description above is the Goldwasser-Micali cryptosystem [16]. Encryption in this system takes place with respect to an RSA modulus n. A '0' bit is encrypted as a quadratic non-residue over Z_n, while a '1' bit is encrypted as a quadratic residue. The key property of this system is its additive homomorphism. In particular, given encryptions $E[b_0]$ and $E[b_1]$ of bits b_0 and b_1 respectively, the Proxy can non-interactively compute $E[b_0 \oplus b_1]$ as $E[b_0]E[b_1]$. Composition of commitments in this manner enables the Proxy to obtain an efficiently checkable proof of correct decommitment from the Sender, as we shall see. We sometimes refer to a Goldwasser-Micali ciphertext as a *quadratic-residue commitment*, abbreviated QR-commitment. We let $E[b]$ denote a Goldwasser-Micali encryption of (QR-commitment to) a bit b. We adopt this notation for clarity, although it is rather loose, disregarding as it does the probabilistic nature of the primitive.

To clarify presentation in the body of the paper, we introduce VPOT through a series of successively refined constructions, beginning with a 1-2 OT protocol involving commitment and then adding on the presence of a Proxy and then the verifiability property.

1.3 Other work on auctions

In consequence of the difficulties involved in deploying standard general secure function evaluation techniques, a number of other secure protocols have been proposed in the literature that are specially tailored for auctions. One of the earliest of these is the scheme of Franklin and Reiter [12]. This scheme is not fully private, in the sense that it only ensures the confidentiality of bids until the end of the protocol (although the authors mention a fully private variant). Some more recent schemes include those of Harkavy, Tygar, and Kikuchi [17], Cachin [5], Sako [24], Di Crescenzo [11], and Jakobsson and Juels [19]. The Harkavy *et al.* scheme is fully privacy preserving, but involves intensive bidder involvement [17], and is not easily adaptable to different auction types or to related protocols. The scheme of Cachin involves two servers, and requires some communication among bidders. At the end of the protocol, a list of bidders is obtained, but not the bid amounts. The scheme of Di Crescenzo [11] requires no communication between bidders, and has low round complexity, but involves the participation of only a single server. The scheme of Sako [24] works on a different principle from these others, involving opening of bids in what is effectively a privacy-preserving Dutch-style auction. While efficient for small auctions, it involves costs linear in the range of possible bids, and does not allow for extension to second-price and other auction types. The Jakobsson and Juels [19] protocol aims at streamlining general secure multi-party computation for functions that involve intensive bitwise manipulation, of which auction protocols, as mentioned above, are a good example. A very recent protocol is that of Baudron and Stern [1]. This protocol is rather expensive, and involves only a single server, with privacy ensured under the condition that there is no collusion between the auction server and any bidder.

Organization

Section 2 reviews some cryptographic building blocks required for our construction. In section 3, we introduce efficient methods to combine bit commitment with oblivious transfer, and develop in detail the important new VPOT protocol. We show how to apply VPOT to the problem of secure function evaluation in section 4. In section 5, we discuss the motivating example: private auction computations. In the appendix, we briefly provide technical details on the Yao circuit constructions, discuss security objectives and assumptions, and provide an efficiency analysis. Due to space constraints, we do not provide formal modeling or proof outlines in this extended abstract.

2 Building Blocks and Background

We review several standard building blocks for our protocols. Further details regarding these primitives may found in the literature. We let $\in_U$ denote uniform, random selection from a set. Details of the Yao construction may be found in the appendix.

Private channels: We assume the use of private, authenticated channels between all three possible pairings of the Chooser, Proxy, and Sender. The private channel between the Chooser and Sender involves the Proxy as an intermediary, for the sake of protocol simplification. We assume that messages are authenticated in a non-repudiable fashion. We do not devote attention to the cryptographic elements underlying these channels. In practice, private channels may be realized by way of, e.g., the Secure Socket Layer protocol (SSL) with supplementary use of digital signatures.

RSA-based 1-2 OT: Recall from above that the aim of 1-2 OT is for the Chooser to obtain a tag t_b for $b \in \{0, 1\}$ from the Sender, who possesses the pair of tags (t_0, t_1). The Chooser should not learn t_{1-b}, and the Sender should not learn b. Most of the proposed practical 1-2 OT protocols in the literature rely on use of El Gamal encryption or some close variant. As an example, we describe the proxy oblivious-transfer protocol of NPS in detail at the beginning of section 3.

In this paper, we introduce a special, RSA-based 1-2 OT protocol. We do not make direct use of the RSA cryptosystem as such in the construction of this primitive. We do, however, employ the familiar RSA setup [23], which we briefly review here. An RSA public key consists of an RSA modulus $n = pq$, where p and q are primes, and a public exponent e such that $gcd(e, \phi(n)) = 1$. The corresponding private key d is such that $ed = 1 \bmod \phi(n)$. Our protocols involve exclusive knowledge and use of a private RSA key d by the Sender.

As a first step in the 1-2 OT protocol, the Sender must provide the Chooser with double commitments $\mathcal{C}_0 = \mathcal{C}_{k_0}(t_0)$ and $\mathcal{C}_1 = \mathcal{C}_{k_1}(t_1)$ on tags t_0 and t_1 respectively. The Sender additionally selects an integer $C \in_U Z_n^*$, which he sends to the Chooser. The Chooser, wishing to receive tag t_b, chooses an element $x \in_U Z_n^*$. If $b = 0$, the Chooser transmits $(x_0, x_1) = (x^3, Cx^3)$ to the Sender; otherwise, she transmits $(x_0, x_1) = (x^3/C, x^3)$. The Sender checks that $x_1/x_0 = C$. If so, he uses his private key to construct $(z_0, z_1) = (x_0^{1/3}k_0, x_1^{1/3}k_1)$, which he sends to the Chooser. The Chooser then makes use of x to extract k_b in the obvious fashion. Given k_b, the chooser can extract t_b from $\mathcal{C}_b$ as desired.

Lacking knowledge of the cube root C, the RSA assumption implies that the Chooser cannot obtain k_{1-b}. In the random oracle model, then, it may be seen that t_{1-b} is hidden from the Chooser in a semantically secure manner. As the Sender does not know for which element in the pair (x_0, x_1) the Chooser possesses the corresponding cube root, it may be seen that b is hidden in an information-theoretic sense from the Sender. Our VPOT protocol is essentially a variant on this basic 1-2 OT scheme.

As noted above, our choice of RSA for our protocols stems from a desire to render computation by the Chooser (corresponding to the bidder in an auction protocol) as efficient as possible. We employ $e = 3$, a common choice, in order to render these computations as rapid as possible, although none of our results depends on this fact.

Yao Circuit Evaluation: As discussed above, Yao circuit evaluation serves as the cornerstone of our VPOT protocol, as it does for NPS. Informally, the Yao construction encrypts an entire boolean function, using ciphertexts to represent the 0 and 1's in a table composing a "boolean gate". It is easy to see how any function with finite domain and range can be compiled into a *circuit*, namely a finite set of interdependent boolean gates. Construction of Yao circuits is conceptually straightforward for auction functions, which incorporate a collection of '>' comparisons. We present details on Yao circuit construction in the appendix.

Goldwasser-Micali encryption: The concept of probabilistic encryption was introduced [16] and elaborated on in [4] to set forth the notion of semantic security in an encryption scheme. The basic scheme employs a Blum integer $n = pq$; this is the product of two primes, where each prime is congruent to $3 \bmod 4$. (To facilitate our security proofs, we assume that the Blum integer employed here is not the same as the RSA modulus employed for 1-2 OT. In practice, and to simplify our protocol descriptions, we believe that use of the same modulus in both cases is acceptable.) The two primes constitute the private decryption key. Encryption is bitwise: a '0' bit is encoded as a square modulo n, and a '1' bit as a non-square modulo n with Jacobi symbol 1. In other words, the quadratic residuosity (QR) of a ciphertext indicates the value of the plaintext bit.

Knowledge of p and q enables efficient determination of the quadratic residuosity of an element in Z_n.

The Goldwasser-Micali encryption scheme can be employed straightforwardly as a commitment scheme for a player that does not know the factorization of n. To decommit a commitment C_b as a '0' bit, a player provides a square root of C_b modulo n; to decommit as a '1' bit, the player provides a square root of $-C_b$ modulo n. It is easy to see that the scheme is unconditionally binding. Privacy is reducible to the so-called *quadratic residuosity assumption*. Recall from above that a key element of this encryption scheme, and indeed, our reason for employing it, is its useful additive homomorphism: $E[b_0]E[b_1] = E[b_0 \oplus b_1]$. We use this to prove the value of the XOR of two committed bits without revealing any additional information about the individual values of the bits themselves.

3 Verifiable Proxy Oblivious Transfer

As a basis for comparison, we begin by presenting details of the NPS proxy oblivious transfer protocol, whose intuition we sketched above. In an initialization process, the Chooser and Sender agree on a cyclic group G of order w over which computation of discrete logarithms is hard and an associated generator g, as well as a random value $C \in_U G$ whose discrete log is unknown to any player. As usual, we let b denote the choice of the bidder, the pair (t_0, t_1), the tags held by the Sender. The protocol is as follows [20]:

1. The Chooser selects a private key $x \in Z_w$, and computes a pair $(PK_b, PK_{1-b}) = (g^x, C/g^x)$, and sends PK_0 to the Sender via the Proxy. She sends x to the Proxy.
2. The Sender computes $PK_1 = C/PK_0$. The Sender computes the pair $(z_0, z_1) = (E_{PK_0}[\rho(t_0)], E_{PK_1}[\rho(t_1)])$, where E_{PK_i} denotes El Gamal encryption under public key PK_i and ρ denotes a suitable error-detection function. The Sender transmits the pair (z_0, z_1) to the Proxy in a random order.
3. The Proxy attempts to decrypt both values in the pair using x. The Proxy knows he has obtained t_b when the error-detection function ρ shows that the decryption is successful.

It may be observed that provided there is no collusion, neither the Sender nor the Proxy can learn b. It may be shown that under the Decisional Diffie-Hellman assumption, even if the Proxy and Chooser collude, they cannot learn both t_0 and t_1. The weakness in this protocol, hinted at above, is the fact that the Sender can choose to send $t_{b'}$ for b' of its choice simply by transmitting $(z_0, z_1) = (E_{PK_0}[\rho(t_{b'})], E_{PK_1}[\rho(t_{b'})])$. Even with the additional apparatus involved for the NPS auction protocol, neither the Chooser (i.e., a bidder) nor the Proxy (i.e., the auctioneer) can detect this tampering, which permits arbitrary alteration of bids.

We are now ready to remedy this problem by introducing our VPOT protocol. For clarity of presentation, we build up to the full VPOT description in stages. These stages correspond roughly to the addition of a commitment, a proxy, and then a verifiability property. Due to space constraints, we provide only informal descriptions of the security properties of each of these protocols.

3.1 Committed Oblivious Transfer (COT)

Committed oblivious transfer (COT) is a natural fusion of bit commitment and 1-2 oblivious transfer. The utility of tag commitment is evident when such tags determine the behavior in subsequent protocol steps. Some existing solutions to this problem include a protocol by Crépeau [10] that combines black-box bit commitment with oblivious transfer by structuring multiple commitments to have a special XOR relationship. Other solutions may make commitments that are of a special form, such that the Sender may send, along with his encrypted tags, a proof in zero knowledge that the tags he encodes correspond to his commitments.

Terminology in the literature varies, so by committed oblivious transfer (COT) we mean that the Sender commits his two tags to the Chooser prior to expression of the Chooser's selection. Our solution is straightforward, and is based on the RSA-based 1-2 OT protocol described above. Note that in COT, no cube roots need ever be extracted by the Chooser, who therefore need perform only a small amount of computation. Secondly, as mentioned in section 1, we employ a form of "double commitment" here. A commitment here consists of the hash of a cube over the RSA modulus.

This two-party protocol involves a Sender and a Chooser. As part of the setup, we assume that the Sender has a valid RSA key $n = pq$. We suppose without loss of generality that $e = 3$ (where $ed = 1 \bmod lcm(p-1, q-1)$). We make use of a hash function $H : Z_n^* \rightarrow \{0,1\}^l$ for security parameter l. (Note that proof of the security of our construction requires application of the random oracle model to H.) Recall that in our notation, for $k \in Z_n^*$ and $t \in \{0,1\}^t$, we define the double commitment $\mathcal{C}_k(t) = (H(k^3), H(k) \oplus t_0)$. The protocol is as follows:

1. The Sender chooses his desired tags $t_0, t_1 \in_U \{0,1\}^l$, as well as keys $k_0, k_1 \in_U Z_n^*$, and sends two double commitments $\mathcal{C}_0 = \mathcal{C}_{k_0}(t_0)$ and $\mathcal{C}_1 = \mathcal{C}_{k_1}(t_1)$ and a random value C to the Chooser.
2. The Chooser chooses $x \in_U Z_n^*$, and computes x^3. If she wishes to receive the first tag she computes $x_0 = x^3$; otherwise she computes $x_0 = x^3/C$. She sends x_0 to the Sender.
3. On receiving x_0, the Sender computes $x_1 = Cx_0$. Using the private key d, he computes $y_0 = x_0^{1/3}$ and $y_1 = x_1^{1/3}$. He sends the pair $(z_0, z_1) = (y_0 k_0, y_1 k_1)$ to the Chooser.
4. The Chooser first computes the cube of both z_0 and z_1 and checks that $H(z_0^3/x_0)$ and $H(z_1^3/x_1)$ are equal to the first element in $\mathcal{C}_0$ and $\mathcal{C}_1$ respectively. If she has chosen the first tag, she extracts $k_0 = z_0/x$; if she has chosen the second, she extracts $k_1 = z_1/x$. She subsequently decommits t_0 or t_1, as desired.

Security Features:

- The Sender does not learn for which element of (x_0, x_1) the Chooser knows the cube root, so the Sender learns nothing about the choice of the Chooser. This holds in an information-theoretic sense.
- Under the RSA assumption, the Chooser cannot feasibly compute the cube root of C, so she cannot obtain the cube roots of both elements of (x_0, x_1). In the random oracle model, the unselected tag is hidden in a semantically secure sense from the Chooser.

- The Chooser knows from the check in the last step that the Sender has correctly transmitted both t_0 and t_1, even though she can extract only one of them. In particular, under the RSA assumption and in the random oracle model, it is infeasible for the Sender to transmit a tag incorrectly without detection.

3.2 Committed Proxy Oblivious Transfer (CPOT)

Committed Oblivious Transfer may be extended to permit a Proxy to receive the tag selected by the Chooser. We call this enhanced protocol Committed Proxy Oblivious Transfer (CPOT). The security properties desired for the Chooser and Sender are as in COT. For the Proxy, we want to ensure two privacy properties. First, even though the Proxy receives a tag on behalf of the Chooser, the Proxy should be unable to determine *which* tag he received. Second, the Proxy should not learn the tag that was not selected by the Chooser. To ensure that the Proxy does not learn which tag he received, the tags are sent in a random order. A final property we seek in CPOT is the ability of the Proxy to verify that the Sender has indeed sent both t_0 and t_1. The setup is as in COT; the protocol is as follows:

1. The Sender chooses his desired tags $t_0, t_1 \in_U \{0,1\}^l$, as well as keys $k_0, k_1 \in_U Z_n^*$, and sends two double commitments $\mathcal{C}_0 = (H({k_0}^3), H(k_0) \oplus t_0)$ and $\mathcal{C}_1 = (H({k_1}^3), H(k_1) \oplus t_0)$ to the Proxy *in a random order*, along with and a random value $C \in_U Z_n^*$.
2. The Proxy forwards C to the Chooser.
3. The Chooser splits b uniformly at random into bits b_P and b_S such that $b = b_P \oplus b_S$. She also selects $x \in_U Z_n^*$. If $b_P = 0$ he computes $x_0 = x^3$; otherwise, she computes $x_0 = x^3/C$. She also computes $v = E[b_S]$.
4. The Chooser sends (x_0, x) to the Proxy. She also sends (x_0, v) to the Sender (via the Proxy, if desired).
5. The Sender receives x_0, computes $x_1 = Cx_0$ and then $y_0 = x_0^{1/3}$ and $y_1 = x_1^{1/3}$. He decrypts b_S. If $b_S = 0$, the Sender transmits the pair $(z_0, z_1) = (y_0 k_0, y_1 k_1)$ to the Proxy; otherwise he transmits the pair $(y_0 k_1, y_1 k_0)$.
6. The Proxy first computes the cube of both z_0 and z_1 and checks that $H(z_0^3/x_0)$ and $H(z_1^3/x_1)$ match the first elements in $\mathcal{C}_0$ and $\mathcal{C}_1$ respectively. If x is the cube root of x_0, he is able to extract $k_b = z_0/x$ and then decommit t_0; otherwise, he extracts $k_b = z_1/x$ and decommits t_1.

Security Features:

- The Sender learns no information about b_P, and, under the quadratic residuosity assumption governing the security of E, the Proxy does not learn b_S. It follows that the Sender or Proxy individually cannot determine the choice b of the Chooser.
- The Proxy cannot feasibly compute the cube root of C under the RSA assumption, and therefore cannot learn the cube roots of both x_0 and x_1. In the random oracle model, thereofer, the unselected tag is hidden in a semantically secure sense from the Proxy. Observe that this is true even if the Proxy cheats or colludes with the Chooser.
- The Proxy can verify that the Sender has correctly transmitted both t_0 and t_1, even though he can extract only one of them.
- Note, however, that neither the Chooser nor the Proxy knows if the Sender has cheated by reversing the roles of t_0 and t_1, or equivalently, flipping the bit b_S.

This last property is similar to the weakness present in the NPS proxy oblivious transfer protocol, namely the one that permits the Sender (auction issuer) to tamper with bids. VPOT, as we shall now see, solves this problem by having the Sender add an efficient proof that he has not reversed the order of the tags.

3.3 Verifiable Proxy Oblivious Transfer (VPOT)

The protocol described above, CPOT, has the important feature that the Sender is forced to transmit both t_0 and t_1 through the OT protocol. As explained, though, he may "flip the bit" b, i.e., send the tag t_{1-b}, rather than the tag t_b requested by the Chooser.

VPOT detects this sort of cheating though use of a zero-knowledge proof based on QR-commitment. The intuition is this: the Sender provides a pair $(\mathcal{C}_0, \mathcal{C}_1)$ of commitments to the tags t_0 and t_1, in a random order. The Sender also commits to an ordering a of these commitments. In particular, $a = 0$ if $\mathcal{C}_0$ represents a commitment to t_0 (and $\mathcal{C}_1$ represents a commitment of t_1); otherwise, $a = 1$. The Sender provides this bit a for the Proxy as a QR-commitment of the form $E[a]$. As in CPOT, the Sender obtains a share b_S of b, the ciphertext $E[b_S]$ being observed by the Proxy. The Proxy therefore can compute a commitment to the bit $c = a \oplus b_S$; in particular, $E[c] = E[a]E[b_S]$. If the Sender provides correct decommitment information, the value c will specify whether the Proxy should be able to open $\mathcal{C}_0$ or $\mathcal{C}_1$. In particular, if $c = 0$, the Proxy should be able to open $\mathcal{C}_0$; otherwise the Proxy should be able to open $\mathcal{C}_1$. To prove correct behavior, the Sender decommits c for the Proxy by proving the quadratic residuosity of $E[c]$. Observe that the bit c, since it is "masked" by the secret bit a, does not reveal information about b_S to the Proxy. Hence the Proxy does not learn the bit b specifying the tag requested by the Chooser.

The following is the full VPOT protocol.

1. The Sender chooses his desired tags $t_0, t_1 \in_U \{0,1\}^l$ and also an integer $C \in_U Z_n^*$.
2. The Sender computes commitments $\mathcal{C}_0 = \mathcal{C}_{k_a}(t_a)$ and $\mathcal{C}_1 = \mathcal{C}_{k_{1-a}}(t_{1-a})$. Let $u = E[a]$, i.e., a QR-commitment of a. The Sender also computes a commitment $CO = H[u]$ to ordering of $(\mathcal{C}_0, \mathcal{C}_1)$. The Sender transmits the pair $(\mathcal{C}_0, \mathcal{C}_1)$ to the Proxy, along with CO.
3. The Chooser receives C from the Proxy and splits b uniformly at random into bits b_P and b_S such that $b = b_P \oplus b_S$. She also selects $x \in_U Z_n^*$. If $b_P = 0$ she computes $x_0 = x^3$; otherwise, she computes $x_0 = x^3/C$. She also computes $v = E[b_S]$.
4. The Chooser sends (x_0, v, x) to the Proxy. She also sends (x_0, v) to the Sender (via the Proxy, if desired).
5. The Sender receives x_0 and computes $x_1 = Cx_0$. He then computes $y_0 = x_0^{1/3}$ and $y_1 = x_1^{1/3}$. He decrypts b_S. If $b_S = 0$, the Sender transmits the pair $(z_0, z_1) = (y_0 k_0, y_1 k_1)$ to the Proxy; otherwise he transmits the pair $(y_0 k_1, y_1 k_0)$.
6. The Sender transmits u to the Proxy (undoing the outer commitment in CO). The Sender then reveals $c = a \oplus b_S$ by decommitting $uv = E[a]E[b_S] = E[c]$. The decommitment of uv is provided as a value ρ such that $\rho^2 = uv$ if $c = 0$ and $\rho^2 = -uv$ if $c = 1$. The Proxy checks the correctness of these decommitments.
7. The Proxy first computes the cube of both z_0 and z_1 and checks that $H(z_0^3/x_0)$ and $H(z_1^3/x_1)$ are equal to the first element of $\mathcal{C}_0$ and the first

element of $\mathcal{C}_1$, in either order. As a final step, the Proxy checks that he can use x to open $\mathcal{C}_0$ if $c = 0$ and $\mathcal{C}_1$ if $c = 1$. This check ensures that the Sender decommitted in the correct order.

Critical Additional Property:

- The Proxy can verify that the Sender has correctly sent him t_b for the bit b selected by the Chooser. Assuming that the Sender and Proxy do not collude, therefore, the Chooser can be assured that the Proxy has received the correct tag t_b.

4 Two-Server Secure-Function Evaluation

In this section we describe an architecture for secure computation based on Yao circuits and VPOT. Due to lack of space, we cannot provide formal modeling and proofs for our auction protocol in this paper. We briefly describe the security requirements informally in the appendix. As above, we assume the availability of private, authenticated channels among participating players.

4.1 Putting together VPOT and Yao circuits

We now combine the VPOT protocol with Yao circuit to construct a secure function evaluation protocol involving two servers (evaluators) and multiple contributors of input values. For consistency with our protocol descriptions above, we refer to the two servers as the Proxy and the Sender. Our secure-computation protocol is designed to evaluate functions on inputs contributed by an arbitrarily large number m of players. We refer to these players as Choosers.

Our aim is to evaluate a function F on the m inputs of the Choosers. The Proxy and Sender together evaluate and publish the result of the function computation, and also provide each player with a receipt to guarantee correctness. The role of the Sender here is to construct Yao circuits and that of the Proxy, to evaluate these circuits. To compute input tags for the Yao circuits, these servers must process separate, parallel invocations of VPOT for every individual bit.

The complete function evaluation protocol is as follows:

Offline Steps

1. The Sender generates an RSA modulus n and publishes this for the VPOT invocations in the current function evaluation session. (**Note**: It is in fact critical that a new RSA modulus be published for each session so as to ensure the privacy properties of VPOT across sessions.)
2. The Sender constructs N copies of Yao circuits to evaluate the function F. He sends these circuits to the Proxy, with double commitments to the garbled input tags, as in VPOT. He also publishes a lookup hash table enabling Yao-output-to-plaintext translation.
3. The Proxy selects half of the circuits at random, and asks the Sender to "open" them.
4. The Sender "opens" the selected circuits and sends the keys to all of their committed garbled input tags. This enables verification of their correct construction. Note that this constitutes half of a cut-and-choose proof, the remaining half involving verification of consistent output on the unopened circuits.

5. The Proxy verifies that the “opened” circuits and committed input tags do indeed calculate the correct function.

VPOT steps

1. The Choosers submit their inputs bitwise to the Proxy according to the VPOT protocol.
2. The Proxy forwards these choices to the Sender according to the VPOT protocol.
3. The Sender sends the garbled input tags according to VPOT for each input bit, and also each of the $N/2$ unopened circuits.
4. If either the Proxy or Sender detects the presence of an ill-formed input by a Chooser, this is proven to the other server. Together the two servers can annul the input of any Chooser, provided that F is suitably constructed. Details are straightforward, and omitted here.

Circuit Evaluation

1. The Proxy checks the garbled tags against the commitments, and evaluates the unopened $N/2$ Yao circuits.
2. The Proxy looks up the Yao circuit outputs in the lookup tables, and verifies that the results of the $N/2$ trials are identical.
3. The Proxy publishes the output entries of the Yao tables, along with the function output. If the entries and output are correct, then the Sender certifies the output.

We remark that the Proxy should not publish the garbled output strings if the outputs are inconsistent. Such a situation only occurs if the Sender cheats, and revealing the outputs might leak some information about input values. Once the correct output values are published, the result will be verifiable by any outside party.

5 Application to Auctions

The two-server secure-function evaluation scheme presented in the previous section can be applied straightforwardly, of course, to create a sealed-bid auction system. As auctions are our key motivating application for the work in this paper, it is worth a brief, concluding discussion of the particular benefits of our approach to auctions.

As explained above, our scheme in this paper addresses the flaw in the NPS auction protocol [20]. The NPS protocol is privacy preserving, but, as already mentioned, effectively operates (unbeknownst to the authors) under the assumption that both the servers are honest. The flaw in this paper is simple: the sender may flip or set constant the two tags which he sends in the oblivious transfer for a given bit, e.g., he can send only ’0’ tags for a given bit submitted by a bidder. This allows the Sender to change the bid of any bidder to any value that the Sender chooses. Nonetheless, we believe that NPS offer a key insight in suggesting a two-server model to exploit the potentially high computational efficiency and low round complexity of the Yao construction. This insight represents an important step toward the realization of practical, privacy-preserving auction protocols.

The secure-function evaluation procedure that we propose in section 4 not only fixes the flaw in the NPS protocol but, as already noted, has the additional benefit of substantially reducing the computation required by bidders. In summary, then, our proposed architecture offers the following features:

1. *Non-interactivity:* Bidders submit bids in a non-interactive fashion. That is, they present their bids to the servers, but need not participate subsequently in the auction protocol except to learn the outcome.
2. *Auction adaptability:* Our auction protocol is readily adaptable with little overhead to a range of auction types, such as highest-price auctions and Vickrey auctions.
3. *Full privacy:* We characterize privacy in terms of a static, active adversary that controls at most one server and an arbitrary number of bidders. The only information revealed to such an adversary at the conclusion of the protocol about the bids of any honest bidders is the outcome of the auction. In a highest-price auction, for example, such an adversary learns only the winning bid and the identity of the winning bidder.
4. *Correctness:* Any player can be assured of the correctness of the auction execution assuming that the two auction servers do not collude.
5. *Robustness:* While we do not achieve robustness against failure by either of the auction servers, the servers can eliminate any ill-formed bids and process the remaining ones correctly.
6. *Low round-complexity:* The protocol involves only five communication passes; this includes the offline cut-and-choose proof of correct Yao circuit construction.
7. *High computational efficiency:* Our protocol is highly efficient in terms of computational requirements. For bidders, it is more so than any other cryptographically based privacy-preserving auction scheme in the literature. The requirement for a bidder in a typical auction would be several tens of modular multiplications (as opposed to a comparable number of modular *exponentiations* in NPS). The cost for the servers is about twice that in NPS. (While in general it is desirable to shed server load in favor of computation on the part of clients, the NPS protocol is so computationally intensive for clients as to pose a likely bottleneck even for reasonably powerful handheld devices.)

The principal drawback of our scheme is that, like the NPS protocol, it does not extend to a trust model involving more than two servers. Whether or not the basic NPS scheme can incorporate multiple servers is an open research question.

We wish to emphasize that, given the succesful implementation experiments of NPS, our proposed architecture is likely to be amenable to very practical deployment in software. Thus we believe that our scheme holds good practical promise. With this in mind, we provide a brief efficiency analysis in the paper appendix.

References

1. O. Baudron and J. Stern. Non-interactive private auctions. In S. Haber, editor, *Financial Cryptography '01*, pages 303–313, 2001.
2. D. Beaver. Minimal-latency secure function evaluation. In B. Preneel, editor, *Advances in Cryptology - Eurocrypt '00*, pages 335–350. Springer-Verlag, 2000. LNCS no. 1807.
3. M. Bellare, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *ACM CCS '90*, pages 503–513. ACM Press, 1990.
4. M. Blum and S. Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In G.R Blakely and D. Chaum, editors, *Advances in Cryptology - Crypto '84*, pages 289–299. Springer-Verlag, 1985. LNCS No. 196.
5. C. Cachin. Efficient private bidding and auctions with an oblivious third party. In G. Tsudik, editor, *ACM CCS '99*, pages 120–127. ACM Press, 1999.

6. C. Cachin and J. Camenisch. Optimistic fair secure computation. In M. Bellare, editor, *Advances in Cryptology - Crypto '00*, pages 94–112. Springer-Verlag, 2000. LNCS no. 1880.
7. C. Cachin, J. Camenisch, J. Kilian, and J. Muller. One-round secure computation and secure autonomous mobile agents, 2000.
8. R. Cramer, I. Damgård, and J.B. Nielsen. Multiparty computation from threshold homomorphic encryption. In B. Pfitzmann, editor, *Advances in Cryptology - Eurocrypt '01*, pages 280–300. Springer-Verlag, 2001. LNCS no. 2045.
9. Claude Crepéau. Verifiable disclosure of secrets and applications. In J.J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology - Eurocrypt '89*, pages 181–191. Springer-Verlag, 1990. LNCS no. 434.
10. Claude Crepéau, van de Graaf, Jeroen, and Alain Tapp. Committed oblivious transfer and private multi-party computation. In D. Coppersmith, editor, *Advances in Cryptology - Crypto '95*, pages 110–123. Springer-Verlag, 1995. LNCS No. 963.
11. G. Di Crescenzo. Private selective payment protocols. In P. Syverson, editor, *Financial Cryptography '00*, 2000.
12. M. Franklin and M. Reiter. The design and implementation of a secure auction server. *IEEE Transactions on Information Theory*, 22(5):302–312, 1996.
13. M. Franklin and M. Yung. Varieties of secure distributed computing. In *Proc. Sequences II, Methods in Communications, Security and Computer Science*, pages 392–417, 1991.
14. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.
15. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87*, pages 218–229. ACM Press, 1987.
16. S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comp. Sys. Sci*, 28(1):270–299, 1984.
17. M. Harkavy, J.D. Tygar, and H. Kikuchi. Electronic auctions with private bids. In *3rd USENIX Workshop on Electronic Commerce*, pages 61–73, 1999.
18. M. Hirt, U. Maurer, and B. Przydatek. Efficient secure multi-party computation. In T. Okamoto, editor, *Advances in Cryptology - Asiacrypt '00*, pages 143–161. Springer-Verlag, 2000. LNCS No. 1976.
19. M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In T. Okamoto, editor, *Advances in Cryptology - Asiacrypt '00*, pages 162–177. Springer-Verlag, 2000. LNCS No. 1976.
20. M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *1st ACM Conf. on Electronic Commerce*, pages 129–139. ACM Press, 1999.
21. M. Rabin. How to exchange secrets by oblivious transfer, 1991. Tech. Memo TR-81 Aiken Computation Laboratory, Harvard University.
22. R. L. Rivest. Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer, 1999.
23. R. L. Rivest, A. Shamir, and L. M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1977.
24. K. Sako. An auction protocol which hides bids of losers. In H. Imai and Y. Zheng, editors, *PKC '00*, pages 422–432. Springer-Verlag, 2000. LNCS no. 1751.
25. A.C. Yao. Protocols for secure computations (extended abstract). In *FOCS '82*, pages 160–164. IEEE Computer Society, 1982.

A Constructing Yao Circuits

Yao circuit construction: We now elaborate on the "garbled" circuit construction of Yao [25] and Goldreich *et al.* [15] sketched above. We employ the notation of [20]. Recall that the goal of player A in constructing such a circuit is to enable player B to be able to evaluate a function $F(x_A, x_B)$ on a secret input x_B without learning anything about the secret input x_A of player A, apart from the output of F. The idea behind the construction is to represent F as a circuit composed of logical gates computing binary operators such as AND or NOT. Each such gate is represented as a small logical table; each bit value in a table is represented by a random-valued tag of length k bits, where k is a security parameter. (In our construction, tags are drawn from Z_n so as to fit with our 1-2 OT protocols.) Player B evaluates this representation of F by performing lookups in the logical tables. The output of one logical table, represented as a tag, serves as input to the next logical table, according to the architecture of the circuit. Indeed, we may think of tags as traveling along "wires" connecting the logical gates of the circuit. Output tags are translated into bit values according to a decoding table accompanying the Yao circuit representation.

Let us consider a gate that computes the binary logical operator g. Let b_i and b_j denote the true (hidden) bit values on the two input wires i and j of some binary gate and $b_l = g(b_i, b_j)$ be the true (hidden) bit value of the output wire for the gate. To hide these values, the circuit constructor selects binary permutations $\pi_i : b_i \rightarrow c_i$, $\pi_j : b_j \rightarrow c_j$, and $\pi_l : b_l \rightarrow c_l$. The effect of these permutations is to map the input bit values b_i, b_j and the output bit b_l into respective random bit values c_i, c_j, and c_l. In principle, if supplied with a logical table on the garbled bit values, a circuit evaluator can now evaluate the gate using the "garbled" bit values c_i and c_j and obtain a garbled output bit c_l.

The problem with this simple form of garbling is that the evaluator can flip the garbled input bit values c_i and c_j, yielding a false evaluation of the circuit and thereby potentially extracting information that should remain concealed. Thus, an additional component of the full garbling scheme for the gate in question is the inclusion of six random tags $K_i^{(b_i)}, K_j^{(b_j)}$, and $K_l^{(b_l)}$ for $b_i, b_j, b_k \in \{0, 1\}$. Here, $K_i^{(0)}$ represents a '0' bit assignment to b_i, $K_i^{(1)}$ represents a '1' bit assignment to b_i, etc. In the full garbling scheme, the bit value b_i on wire i is represented by a pair $< K_i^{(b_i)}, c_i >$, and likewise for b_j and b_l.

It remains to show how to construct a garbled lookup table for the gate. This lookup table contains four rows, each of the form:

$$(c_i, c_j) : E_{<K_i^{(b_i)}, c_i>, <K_j^{(b_j)}, c_j>}[< K_l^{(g(b_i, b_j))}, c_l >],$$

where E is a symmetric encryption function of some appropriate form (typically XORing the plaintext with $< K_i^{(b_i)}, c_i > \oplus < K_j^{(b_j)}, c_j >$). The idea here is that the tags $K_i^{(b_i)}$ and $K_j^{(b_j)}$ for bit values b_i and b_j "unlock" the tag for the corresponding output bit of the gate, namely $b_l = g(b_i, b_j)$. The reader may easily see how evaluation proceeds in this construction, and also how the construction generalizes to all full boolean circuit. It is also straightforward to show that the underlying bit values remain hidden to an evaluator; privacy is determined by the security parameter k determining the length of the tags.

The only remaining piece of the construction is the initialization. In particular, we must show how the evaluating player B obtains the correct tags for the bits composing x_b that are fed to the input gates of the circuit. To ensure full privacy, player A must provide these tags to player B so as to achieve two properties: First, the value x_B must remain hidden from player A; second, player B must obtain *only* the correct tags for the bit values of x_B. Both of these aims can be accomplished by having player A send tags for a given input wire to player B by way of a 1-2 OT. Say, for instance, that player B wishes to obtain tag $K_i^{(b_i)}$ for his secret input bit b_i. Player A simply engages with player B in a 1-2 OT on the pair of tags $< K_i^{(0)}, K_i^{(1)} >$ in the obvious manner. (Yao [25] demonstrated a 1-2 OT procedure based on the hardness of factoring, while Goldreich *et al.* demonstrated one based on any trapdoor one-way permutation.)

Choose-and-cut proof of correct Yao circuit construction: Cut-and Choose-style proofs employ a strategy of "spot checking". In the NPS auction paper, the authors introduce a simple cut-and-choose proof in which player A proves to player B that she has constructed a garbled Yao circuit correctly. Briefly stated, player A sends N independently garbled copies of the Yao circuit for function F to player B. Player B asks player A to decommit a random set of half of the commitments, i.e., to reveal all of the random tags and permutations. Player B verifies that these are correctly constructed and then evaluates the remaining $N/2$ garbled circuits on his secret input value x_B (doing 1-2 OTs independently for each with player A). Provided that the output of all $N/2$ garbled circuits is identical, player B accepts the output as correct. The probability of player A successfully cheating is greater than $1/\binom{N}{2}$ by a probability negligible in k, the bit length of the tags.

B Security Model Considerations

B.1 Security characteristics

We discuss the security in the context of the complete two-server computation protocol. In this paper we omit proof sketches, but rather discuss the security model, the key relevant assumptions, and the security aims.

Our aim is to achieve security within the standard cryptographic model of secure multi-party computation. This involves m players who are assumed to share an authenticated broadcast channel, and an *adversary* with resources polynomially bounded in all security parameters. Our model is somewhat unusual in that there are only two players, the Sender, and the Proxy, performing most of the computation, yet there are multiple players (Choosers) submitting inputs. For this situation it is natural to consider an adversary who may corrupt up to all but one of the Choosers, and either the Proxy or the Sender. in an active fashion, i.e., the adversary gains access to their private information, and may wholly control their behavior. We assume in our security analysis in the appendix that the adversary is *static*, that is, the adversary must choose in advance which players she wishes to corrupt. This model represents our assumption that there is at least one honest Chooser and that the Proxy and Sender do not collude.

In proving security, our aim is to show, by means of standard (although not straightforward) simulator arguments, that that security in VPOT and the secure function evaluation protocol are reducible to a few canonical cryptographic assumptions:

1. The *RSA Assumption.* This is used with the arguments involving quadratic residues modulo a Blum integer. (In fact, this latter depends for security upon the weaker assumption of the hardness of factoring.)
2. The *Quadratic Residuosity Assumption* (QR-assumption). As mentioned above, we consider only RSA moduli that are Blum integers.
3. The *Random Oracle Model* for hash functions.
4. Use of a digital signature scheme resistant, under the Random Oracle Model for hash functions, to existential forgery by an adversary capable of mounting an adaptive, chosen-message attack. This provides the non-repudiably authenticated channels we require.

Due to lack of space for security proofs, we state here only our security objectives:

Privacy: An adversary as described above learns only non-negligible information about the private values of honest players apart from the output of the function F.

Correctness: It is infeasible for the adversary to cause the protocol to output an incorrect result.

Robustness: We do not obtain robustness in the obvious sense that either the Proxy or Sender may refuse to continue at any step, or equivalently be caught cheating. Because the Choosers in the multi-party computation have a non-interactive role, however, an adversary controlling Choosers alone cannot halt the protocol execution. We therefore obtain robustness against the most common and troublesome form of potential denial-of-service attack.

C Efficiency Considerations

In order to permit easy comparison with other protocols, for example that in [20], we summarize here the computational requirements for the participants in our proposed scheme.

C.1 Computational costs for VPOT

The protocol VPOT involves both offline and online calculations; the latter may be considered of more practical relevance. A typical implementation might use a 1024-bit RSA modulus with exponent 3. Disregarding the cost of hash functions computations, which is relatively small, we observe that the Sender must compute seven modular multiplications offline. Online, the Sender must calculate three modular exponentiations. The Proxy has much less computational expense: only five modular multiplications and two modular divisions. Best of all, the Chooser need only calculate five modular multiplications per bit selection. Note that these are the costs for only one invocation of VPOT. A full auction protocol will involve many, of course, as we now consider.

C.2 A typical auction

To provide a flavor of the resource requirements for our proposed architecture, we summarize the computational requirements in a typical auction setting. We omit the negligible cost of hash calculations, and count circuit evaluations as a unit. We remark that there are optimizations applicable to both the arithmetic and circuit evaluation.

In our example there are 10 bidders in the auction, the bids are 10 bits long, and 10 circuits out of 20 remain after the cut-and-choose step. The Sender must create the 20 circuits offline, and he can also calculate 10,000 of his modular multiplications off-line. During the protocol, he must calculate 2000 modular multiplications and 2000 modular exponentiations. The Proxy must evaluate 20 circuits accepting 100 inputs each, calculate 10,000 modular multiplications, and 2000 modular divisions. About half of this effort can be done off-line before bidding commences. Finally, the Choosers (bidders) need only perform at most 50 modular multiplications each in total to construct their bids. We exclude overhead associated with private, authenticated channel establishment, as there are many ways to implement this portion of the protocol, and it is independent of our investigations in this paper.