Development and Evaluation of a Secure, Privacy Preserving Combinatorial Auction

Ben Palmer **Kris Bubendorfer** **Ian Welch**

School of Engineering and Computer Science
Victoria University Wellington,
PO Box 600, Wellington, New Zealand 6140,
Email: {Ben,Kris,Ian}@ecs.vuw.ac.nz

Abstract

The use of electronic auctions as a means of trading goods has increased year after year. eBay has gone from half a million registered users in 1998 to 88 million today. Businesses have also shown interest in using auctions. However, the traditional single good auction as used by eBay lacks the required ability to express dependencies between goods in complex procurement auctions leading to risky bidding strategies and sub optimal allocations. The use of combinatorial auctions, where bidders can place bids on combinations of goods, allows bidders to take advantage of any dependencies and auctioneers to generate optimal allocations of goods. In this paper we introduce a new algorithm for creating a combinatorial auction circuit that can be used to compute the result of a combinatorial auction by any garbled circuit auction protocol. In an electronic auction bids from competing parties are commercially sensitive information as bidders will not want their competitors finding out the value they place on a given item. Therefore, there has been considerable research into auction protocols that protect knowledge of all bids except the winning bid from everyone, including the auctioneer. The Garbled Circuit (GC) protocol as described by Naor, Pinkas and Sumner is an example of such an auction. However, it has only been used to provide privacy for single good auctions rather than combinatorial auctions and has been considered impractical for realistically sized auctions due to the protocol's communication overheads. Using our algorithm for creating combinatorial auction circuits, the GC protocol can conduct combinatorial auction while keeping losing bid values secret. We have also conducted performance measurements on both the computation and communication overhead of the GC protocol using our combinatorial auction circuit. These experiments show that the communication overhead is low enough to allow its use for realistically sized auctions (6MB for an auction with 3 goods, a maximum price of 16, and 100 bidders).
Keywords: E-Commerce, Distributed Systems, Security.

1 Introduction

Electronic markets such as eBay and Trade Me have changed the way people buy and sell goods online. eBay and Trade Me support fixed price (the buy now button) and standard auctions using an open outcry English auction protocol. Through the use of such web-based auction services, auctions have become an understood and accepted way for people to trade goods.

A combinatorial auction differs from a normal auction by permitting bidders to express a preference for more than a single good. An arbitrary collection of items defined by the bidder, can have a combined value greater than the sum of the individual items. Bids can be made that are conditional upon obtaining the entire set of desired items. As a simple example, consider a real estate auction, where three adjacent lots (A,B and C) are up for sale. The developer of a retail centre needs a minimum of 2 adjacent lots. If this was treated as 3 separate auctions, the value of lot B (to the developer) would be greater than A or C as winning A or C without B would have no value. The various bidder strategies in this auction are complex, involve risk, and are dependent on the order of the auctions due to the dependencies between the lots. The inability of the single good auction to express such dependencies can lead to sub optimal allocations. A combinatorial auction permits bidders to express these dependencies and thereby enable the auction to result in optimal allocations of goods to bidders.

Garbled circuits are a software technique first presented by Yao (Yao 1982) as a solution to the Millionaire's Problem, in which two millionaires wish to determine who is richer – without revealing their actual wealth to the other. A garbled circuit involves the creation of a set of Boolean gates in software to compute a function, and then the garbling of the circuit to obfuscate the input and intermediate values, but still allow execution of the function. The principle idea of a garbled circuit is to act as a replacement for the trusted party in transactions between mutually distrustful parties.

Trust is a concept that we humans implicitly understand, but have difficulty in applying this understanding digitally. Trust takes into account the role of the entity, the degree of potential loss and sometimes prior experience or experience of those trusted by you. However, trust can be misplaced, and the degree of risk underestimated. A trusted entity is not necessarily trustworthy. This applies to electronic auctions in particular, as the social mechanisms that enforce trustworthy behaviour in traditional auctions are missing.

Imagine the following scenario from (Bubendorfer et al. 2009). Bob and Jane have surplus resources and wish to sell these resources via Alice, their auctioneer. The auction is a sealed bid reverse auction (or tender), where clients issue requests for resources and resource providers bid (and compete) to supply them. Alice's auction house is hosted using resources provided by Sam. When a client submits a resource request to Alice, Alice creates an auction and advertises the new auction to Bob and Jane. Bob and Jane

respond by submitting their bids to Alice. At the end of the auction, Alice examines the bids and declares the winner of the auction.

In this scenario bid privacy can be compromised in a number of ways. Alice can freely examine the bids from Bob and Jane. She can then leak this information to others giving them a competitive advantage. Sam could also obtain this information directly from the memory allocated to Alice, or if it were encrypted, extract Alice's key from memory. If Alice or Sam were also resource providers, then the incentive to cheat is considerable.

One way to solve these problems is to ensure that bids are kept private, that is, hidden from Alice and Sam. At first it seems that this is impossible, as Alice would be unable to compute the winner of the auction. However, we can utilise garbled circuits that enable Alice to compute the outcome of the auction, without revealing anything other than the winner and the price paid. The most notable *single good* garbled circuit auction protocol that utilises this solution is by Naor, Pinkas and Sumner (Naor et al. 1999), other garbled circuit auction protocols include (Jakobsson & Juels 2000, Baudron & Stern 2001, Kurosawa & Ogata 2002). There are no existing auction protocols that utilise this solution for combinatorial auctions.

The contribution of this paper is the creation of a novel algorithm to construct combinatorial auction circuits. The resulting circuit is then used to compute the results of a *combinatorial* auction when given the number of goods, bidders, and the maximum price. Our combinatorial auction circuit can be used with *any single good* privacy preserving auction protocol, based on garbled circuits (Naor et al. 1999, Jakobsson & Juels 2000, Baudron & Stern 2001), to extend it for combinatorial auctions. This is the first example of a combinatorial auction circuit to appear in the literature and we present the circuit and the algorithms used to generate it.

A criticism that is often levelled at garbled circuits is the communication overhead caused by the garbled circuit that is sent from the auction issuer to the auctioneer (Yokoo & Suzuki 2004, 2002, Perrig et al. 2001). Even the creators of the garbled circuit auction protocol state that:

> *As for the communication overhead, the tables that code the circuit can be sent from the AI to the auctioneer in advance, before the auction begins, possibly on a CD-ROM or DVD* (Naor et al. 1999).

The above quote suggests that it may not be feasible to transmit an auction circuit over a network. However, we can show empirically that that it is indeed feasible and this is the case even for multiple good combinatorial auctions. We have implemented the garbled circuit auction protocol by Naor and Pinkas (Naor et al. 1999) and the Verifiable Proxy Oblivious Transfer (VPOT) protocol (Juels & Szydlo 2003) introduced by Juels and Szydlo to fix a problem with the original garbled circuits auction protocol. No performance results have been published for the garbled circuit auction protocol using VPOT before this work. Finally we compare the performance of this protocol with another well known privacy preserving combinatorial auction protocol based on threshold trust.

2 Related Work

There are two main approaches used to ensure the privacy of bidder valuations; threshold trust (Franklin & Reiter 1995, Yokoo & Suzuki 2002, Suzuki & Yokoo 2002, Harkavy et al. 1998, Peng et al. 2002, Bubendorfer & Thomson 2006) and two party trust (Lipmaa et al. 2002, Naor et al. 1999, Juels & Szydlo 2003, Cachin 1999, Kikuchi 2001). In threshold trust, the co-operation of some quorum of hosts is required to reconstruct a bid. Threshold trust is secure as long as the quorum of honest hosts can be met. To implement threshold trust, different protocols have used different techniques. A threshold El-Gamal homomorphic crypto system has been used to allow computation on encrypted bids while needing a quorum of hosts to decrypt the bids (Yokoo & Suzuki 2002). This homomorphic auction protocol is able to conduct combinatorial auctions. Polynomial secret sharing has also been used (Kikuchi 2001) and extended to conduct combinatorial auctions (Suzuki & Yokoo 2002). Threshold trust has been criticised for requiring a heterogeneous collection of hosts from different organisations willing to commit computing resources to host an auction (Lipmaa et al. 2002). It is easier to find two parties from separate organisations willing to conduct an auction for two party trust than to find a larger group of parties to conduct an auction using threshold trust.

Two party trust relies on a symmetric separation of duty between two parties with the information being kept private as long as the two parties do not collude. Garbled circuits are a two party trust protocol (Naor et al. 1999, Juels & Szydlo 2003) that uses an auctioneer and an auction issuer as the two parties. Garbled circuits preserve the communication pattern of traditional auctions in that bidders just send information to the auctioneer, and only the auctioneer sends information to the auction issuer. The bidders in a garbled circuit auction do not have to encrypt bid values, which can be computationally expensive if the bidders are low power devices. The VPOT protocol addresses a security flaw in the original garbled circuit auction protocol by replacing the proxy oblivious transfer protocol with a verifiable proxy oblivious transfer protocol.

A novel auction protocol has been developed where an auctioneer uses a third party to obliviously compare bid values (Cachin 1999) . In this protocol one of the parties learns a partial ordering of the bids, and if the other party colludes with a bidder, then that bidder could see all the comparisons.

A similar auction protocol to garbled circuits that does not use an auction issuer but where instead bidders perform the role of the auction issuer has also been developed (Baudron & Stern 2001). Unfortunately this protocol is restricted to five or six bidders in real world situations and a malicious auctioneer could collude with a bidder to break the assumptions of the protocol.

3 A Combinatorial Auction Circuit

A circuit is a network of Boolean gates with a set of inputs, a set of intermediate gates, and a set of outputs gates. Figure 1 shows a simple worked example of an auction circuit. This circuit can compute the result of an auction with one good, two bidders and uses two bits to represent the prices. The inputs to the circuit are the 2-bit bids from the two bidders. The outputs of the circuit are two Boolean values that indicate whether bidder one or bidder two was the winner and a 2-bit value that is the maximum (winning) price. In our example, when presented with the input values in the figure, bidder one bids `10` and bidder two bids `11`, the circuit computes that bidder two wins the auction with a maximum price of `11`. As a further example let's change the inputs, and keep the same circuit. Bidder one now bids `10` and bidder two now bids `01`.

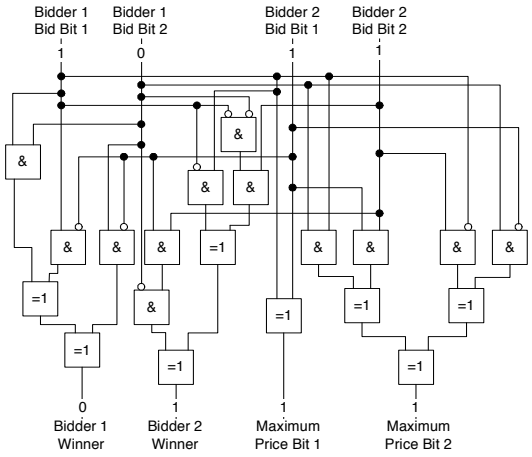


Figure 1: A Simple Auction Circuit. A & gate represents an AND gate and a = 1 gate represents an OR gate. Solid circles represent a join in the wires and unfilled circuits represent NOT gates.

For these inputs, the circuit computes that bidder one wins the auction with a maximum price of 10. Clearly the circuits get more complex with higher numbers of bidders, available prices, and goods.

Auction circuits need to be created dynamically based on the parameters of the auction. An algorithm is needed that can return a Boolean circuit for computing the result of a combinatorial auction taking as inputs the number of bidders, the maximum price, and the number of goods.

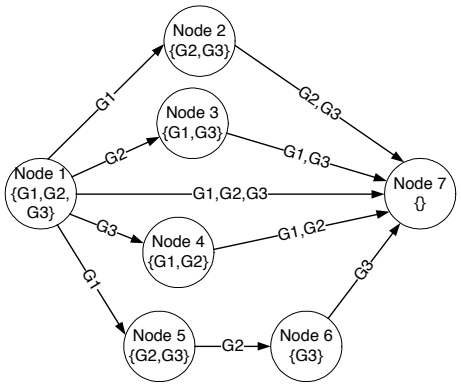
3.1 Building Blocks

We make use of the single good **1st price circuit** of Kurosawa and Ogata (Kurosawa & Ogata 2002) as a building block for our combinatorial auction circuit. A 1st price auction returns the highest bid as the winner. The circuit is constructed of NOT, AND, OR, XOR, and SELECT gates. A SELECT gate has three inputs, if the first input is true it outputs the second input, and if the first input is false it outputs the third input. The single good auction circuit by Kurosawa and Ogata uses a technique they term bit slicing where the bits of the various bids are compared from most significant to least significant. This is in contrast to the standard first price circuit that computes the millionaires problem comparing each bidder's bid in turn. We also use a basic **add circuit** that given two bitwise values as input, outputs the sum of these two values.

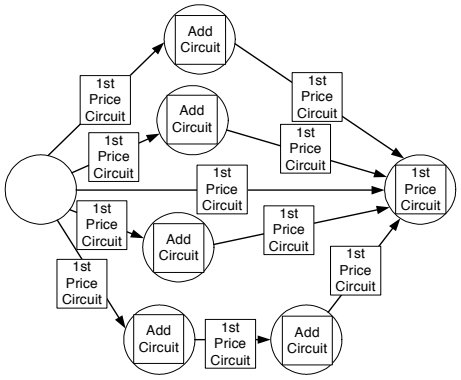
Combinatorial auctions can be represented as an **auction graph** (Figure 2(a)) where nodes represent goods, **links** between nodes represent a subset of goods, and each complete **path** through the graph represents an allocation of the goods. The **optimal path** through an auction graph is the path that returns the highest revenue. The auction graph representation of combinatorial auctions has been used in several previous works (Yokoo & Suzuki 2002, Suzuki & Yokoo 2002).

3.2 The Complete Circuit

The auction graph representation, the 1st price circuit and the add circuit are used to create a circuit to compute the optimal value for a combinatorial auction along with the winning bidders and prices. Figure 2 shows a three good auction graph in 2(a) and 2(b) shows the construction of the resulting combinatorial auction circuit. Every link in the auction graph



(a) Three Good Auction Graph



(b) Auction Circuit Graph

Figure 2: Creating a Combinatorial Auction Circuit based on an Auction Graph

has a 1st price circuit that outputs the maximum bid for that link. Every node in the auction graph except the last node has an add circuit that adds the maximum bid for the incoming link to the bids on the outgoing link. The last node has a final 1st price circuit that outputs the optimal value for the combinatorial auction.

An auction circuit for combinatorial auctions needs to compute and output not only the optimal value for the auction, but also which bidders won which goods and at what price. Each 1st price circuit outputs the maximum bid for that link and the associated bidder. These values are combined for every link in a path by using a SELECT gate to output the winning bidder only if that link is on the optimal path. Further SELECT gates are used to output the winning price for a bidder on a link only if that link is on the optimal path and if that bidder had the maximum price for that link.

3.3 Circuit Creation Algorithm

We now present our algorithm used in the creation of our combinatorial auction circuit. It takes as input the number of bidders, prices, and goods and outputs a circuit for computing the result of a combinatorial auction. The outputs are a series of bits for each bidder that indicate if that bidder won any of the links on the auction graph and the winning bids they need to pay for each link. In case of tie break the circuit outputs both winning bidders and the auctioneer would need to choose some other way of deciding the winner, such as a coin toss.

Algorithm 1 is the main algorithm used in the creation of the circuit. The algorithm is split in to two parts. The first part calculates the optimal path through the auction graph by calculating the maximum bid for each link in the graph, adding together the maximum bids for each path, and then calculating the optimal path based on the maximum bids for each path. The second part of the algorithm calculates the winning bidders and prices using SELECT gates for each bidder and every link and path in the auction graph.

Algorithm 1
Procedure *CreateCombinatorialAuctionCircuit*
Input: nBidders, nPrices, nGoods
Output: AuctionCircuit AC
1. (∗ Create the Auction Graph ∗)
2. AuctionGraph Graph ← CreateAuctionGraph(nGoods)
3. (∗ Loop Through All Paths ∗)
4. **for** Paths $i \in$ Graph
5. (∗ Loop Through Links on Path ∗)
6. **for** Links $j \in i$
7. Create 1st Price Circuit with inputs of the bids for link j
8. (∗ Get the Max Bid for Path i ∗)
9. **if** Number Links on Path $i > 1$)
10. **for** Links $j \in i$
11. AddOutputs(j,j+1,AC)
12. (∗ find the optimal path ∗)
13. Create 1st Price Circuit with Inputs of the Final Add Circuits For Each Path
14. (∗ Now find what bidders won for what price ∗)
15. (∗ Loop Through All Bidders ∗)
16. **for** Bidder $i \in$ nBidders
17. (∗ Loop Through All Paths ∗)
18. **for** Paths $j \in$ Graph
19. (∗ Loop Through Links on Path ∗)
20. **for** Links $k \in j$
21. WinningBiddersPrices (i,j,k,nPrices,AC)
22. return AC

Algorithm 2 is a helper method that is used to add the outputs of the 1st Price Circuit for each link in a path together to get the maximum price for a path in the auction graph. These maximum prices for each path are then compared in the final 1st Price Circuit that outputs the optimal path for an auction.

Algorithm 2
Procedure *AddOutputs*
Input: Link j, Link $j+1$, AuctionCircuit AC
1. **if** (j is the first link in the path)
2. Create Add Circuit with inputs of the maximum bids for link j and $j+1$
3. **else**
4. Create Add Circuit with inputs of the maximum bids for link j+1 and the output of the previous Add Circuit

Algorithm 3 is executed for every bidder and every link in every path of the graph. The first SELECT gate outputs the winning bidder of the 1st Price Circuit for this link in the auction graph provided the link is on the optimal path. The second two SELECT gates output the winning price of the 1st Price Circuit for this bidder and link in the auction graph provided the path is on the optimal path and this bidder was the winner of the link.

Algorithm 3
Procedure *WinningBiddersPrices*
Input: Bidder i, Path j, Link k, nPrices, AuctionCircuit AC

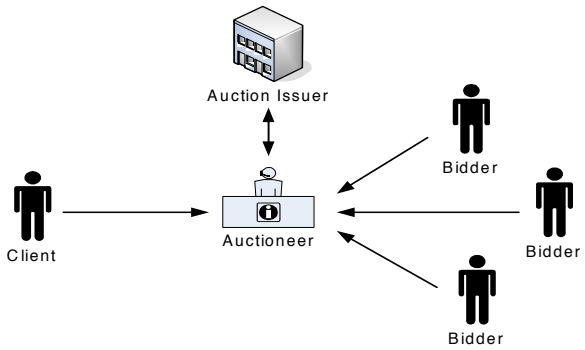


Figure 3: Garbled Circuit Parties

1. (∗ Creates gate to work out the winning bidders and prices for this link in the graph ∗)
2. Create a SELECT gate with 3 inputs. The first input is the output of the final 1st Price Circuit for path j, the second input is the output of the 1st Price Circuit for bidder i and link k, and the third input is the output of the final 1st Price Circuit for path j
3. **for** $m < nPrices$
4. Create a SELECT gate with 3 inputs. The first input is the output of the final 1st Price Circuit for path j, the second input is the winning price for link k at price m, and the third input is the output of the final 1st Price Circuit for path j
5. Create a SELECT gate with 3 inputs. The first input is the output of the 1st Price Circuit for bidder i and link k, the second input is the previous SELECT node, and the third input is the output of the 1st Price Circuit for bidder i and link k

4 Garbled Circuits Auction Protocol

A garbled circuit is a Boolean circuit for computing the result of some function that has been obfuscated by one party to hide the input and intermediate values of the gates of the circuit. When presented with a garbled circuit any party can calculate the result of the function when provided with the garbled input values to the circuit and an output mapping from garbled outputs to the actual output of the original circuit. In the garbled circuit auction protocol, a Boolean circuit is created that outputs the result of the auction (Naor et al. 1999). This circuit is then garbled by a party known as the auction issuer and sent to the auctioneer. Using the garbled circuit created by the auction issuer, the auctioneer is then able to compute the result of the auction after discovering the garbled inputs of the garbled circuit. As long as the auction issuer does not reveal a set of random values it used when garbling the circuit, the input and intermediate values remain hidden from the auctioneer. The verifiable proxy oblivious transfer (VPOT) protocol (Juels & Szydlo 2003) addresses a security flaw in the original garbled circuit auction protocol where the auction issuer could change bids without detection. Figure 3 shows the parties in the garbled circuits auction protocol. The bidders and the client only need to have a connection to the auctioneer, and the auctioneer is the only party that needs a connection to the auction issuer.

The basic steps of a Sealed-Bid auction using the garbled circuit protocol are:

- The client contacts the auctioneer with details of the auction they wish to run.
- The auctioneer advertises details of the auction including the number of goods, number of prices, and the auction issuer being used.
- The auction issuer constructs a garbled circuit for the auction based on how many bidders, goods, and the number of bits in the price as well as a mapping from garbled outputs to outputs.
- The auction issuer sends the garbled circuit and output mapping to the auctioneer.
- The auction issuer, auctioneer, and bidders use a protocol called verifiable proxy oblivious transfer (VPOT) which results in the auctioneer learning the garbled values of the inputs, and the auction issuer and bidders learning no new information.
- The auctioneer executes the garbled circuit using the garbled input and decodes the output using the output mapping sent by the auction issuer.

More details of the garbled circuit auction protocol, including the algorithms used, can be found in the Appendix.

4.1 Security

The security of the garbled circuit auction protocol comes from the garbling of the circuit that is done by the auction issuer. This garbled circuit is then sent to the auctioneer to execute. As long as the auction issuer does not collude with the auctioneer losing bid values are kept secret. During the garbling of the circuit, each wire connecting the nodes in the circuit is assigned a randomly generated value and a randomly generated permutation of the values of the wire that is used to create the garbled value of the wire. A gate table is then created for each node in the circuit that maps the garbled input of the node to the garbled output. A publicly known random function is used to create the gate table ensuring that knowledge of one combination of the garbled inputs of a node does not reveal the other garbled outputs. The VPOT protocol is then executed by the bidders, auctioneer, and the auction issuer after which the auctioneer learns the garbled inputs of the circuit and can execute the circuit to find the garbled outputs. A mapping is provided by the auction issuer that maps the garbled outputs of the circuit to the actual output. Parties in the garbled circuit auction protocol are assumed to be passive adversaries, although in the original paper verification techniques are discussed which can extend the garbled circuit auction protocol to handle active adversaries. A more detailed security analysis of the garbled circuit auction protocol can be found in the original paper (Naor et al. 1999), and a the paper presenting the VPOT protocol contains a detailed analysis of the security of VPOT (Juels & Szydlo 2003).

5 Circuit Size

As stated in the introduction, one of the main criticisms of the garbled circuit auction protocol is the size of the garbled circuit, which is composed of gate tables and an output mapping, that has to be sent from the auction issuer to the auctioneer. Even the creators of the garbled circuit auction protocol suggest sending the gate tables on a CD-ROM or DVD (Naor et al. 1999). In order to investigate these claims, in this section we quantify the size of the gate tables for different combinatorial auctions. We first investigate the complexity of the circuit before providing experimental results on the size of the circuit.

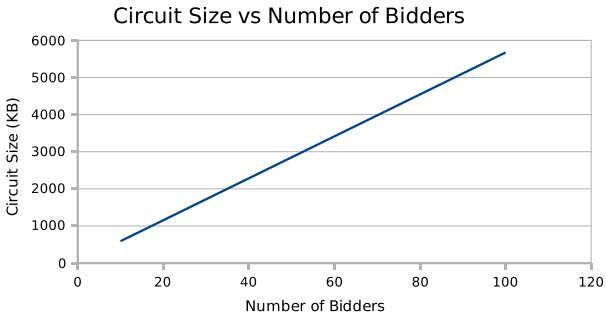


Figure 4: Circuit Size vs Number of Bidders

5.1 Complexity

Table 1 shows the upper bound on the number of gates used in our combinatorial auction circuit where g is the number of goods, b the number of bidders, and p the bits in the price. There are 2^g possible unique combinations of goods. There are B_g possible unique allocations of the g goods where B_g is the Bell Number for the number of goods. For every allocation there can be at most g links in the graph so we assume there are g links for every allocation.

The largest factor influencing the size of the circuit is the number of goods g. Increasing the number of bidders b results in a linear growth in the number of nodes in the circuit. Increasing the number of bits in the price p results in a linear increase in the size of the circuit and an exponential increase in the number of available prices. More available prices mean that bids can be more finely expressed – increasing the bid granularity of the auction protocol. For example, with $p = 4$ there are $16 = 2^4$ available prices but with $p = 5$ there are $32 = 2^5$ possible prices. When the number of goods g is increased, the total number of nodes in the auction circuit increases exponentially. When the number of goods is increased linearly the number of possible combinations of goods increases exponentially as there are 2^g possible combinations of goods.

5.2 Experimental Results

To calculate the size of the garbled circuit, we have taken the number of two input gates in the combinatorial auction circuit and multiplied them by 4 and then by 128. This is because for every two input gate there are 4 entries in the gate table and every entry is 128 bits. The size of the output mapping is not included in this calculation, but is significantly smaller than the size of the gate tables. We have quantified the size of the garbled circuit in respect to the number of bidders, number of goods, and the number of bits in the price. Other than the variable under test, the default parameters selected were ten bidders, three goods, and four bits in the price (for a maximum bid of sixteen).

Figure 4 shows the size of the garbled circuit increasing linearly as the number of bidders increases.

The size of the garbled circuit is proportional to ln(maximum bid) as shown in Figure 5.

Figure 6 shows the size of the garbled circuit (shown on a logarithmic scale) increasing exponentially as the number of goods increases.

The size of the garbled circuits in these tests would not require a CD or DVD to be sent from the auction issuer to the auctioneer and could be sent over the network. For example, an auction with 3 goods, a maximum bid of 16, and 100 bidders has a garbled

No. of Input Nodes	$2^g bp$
No. of Output Nodes	$b(p + (2^g p))$
No. of AND Nodes	$2B_g gp(b + 1)$
No. of OR Nodes	$B_g bg(1 + p) + B_g gp(b + 1)$
No. of SELECT Nodes	$B_g bg(2p + 1) + B_g bgp$
No. of XOR Nodes	$2B_g gp$
Total No. of Nodes	$2^{g+1} bp + B_g gp(4b + 5) + B_g bg(3p + 2) + bp$

Table 1: Number of Nodes in the Auction Circuit

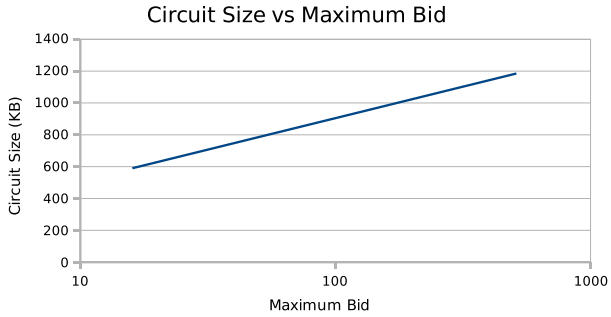


Figure 5: Circuit Size vs Maximum Bid

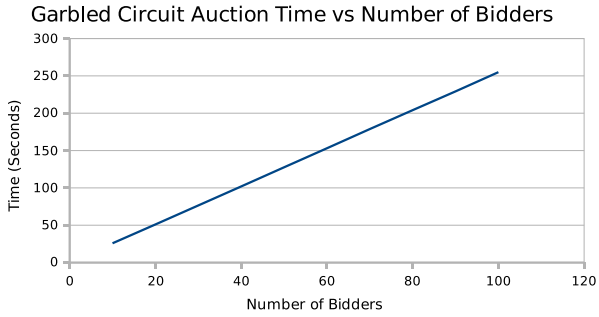


Figure 7: Auction Time vs No of Bidders

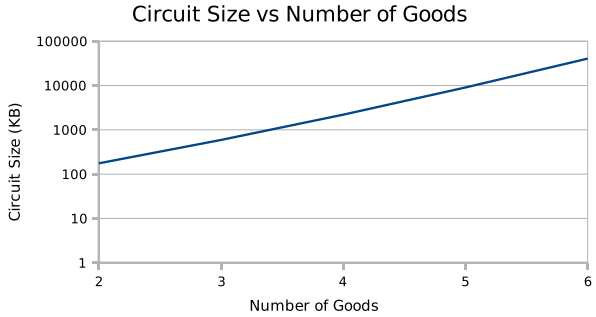


Figure 6: Circuit Size vs No of Goods

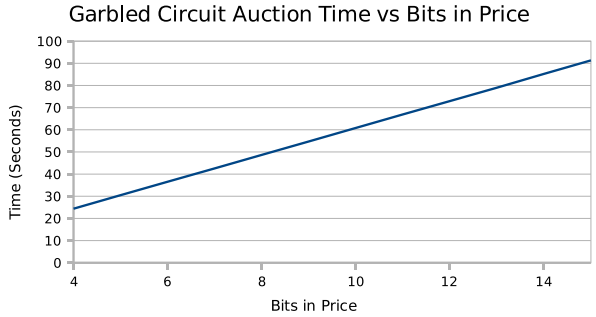


Figure 8: Auction Time vs Bits in Price

circuit of approximately 5MB. A larger auction with 5 goods, a maximum bid of 200, and 50 bidders has a garbled circuit size of approximately 85MB. The size of the garbled circuits gets very large for large number of goods but, if some combinations of goods can be removed as invalid, the garbled circuit size would drop. It is also worth noting that construction of a more compact combinatorial auction circuit with less gates would decrease the size of the garbled circuit.

6 Performance Results

We have tested the performance of the garbled circuits protocol in respect to the number of bidders, number of goods, and the number of bits in the price. Other than the variable under test, the default parameters selected for performance measurements were ten bidders, three goods, and four bits in the price. The test machines were a group of four Dell Optiplex GX755s each with an Intel Core 2 Duo processor and 2048MB DDR SDRAM. The auction time recorded is the total time to compute the auction, this includes the creation of the circuit, the garbling of the circuit, the VPOT protocol to learn the garbled inputs, and the garbled circuit execution time.

The time taken to complete the auction increases linearly as the number of bidders increases as shown in Figure 7. This is due to the linear growth in the time taken to execute the VPOT protocol and to garble and execute the circuit as the number of bidders increases.

Figure 8 shows the time taken to complete the auction when the number of bits in the price is increased. The relationship between the number of bits in the price and the time taken appears to be linear. Increasing the number of bits in the price by 1 bit increases the maximum bid by a power of 2. Figure 9 illustrates the relationship between the time taken to compute the auction and the maximum bid.

Figure 10 compares the performance of garbled circuits with the performance (Bubendorfer & Thom-

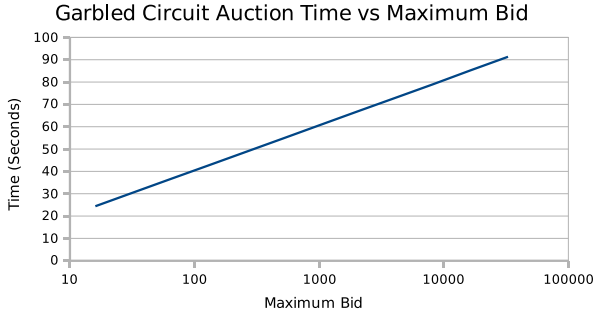


Figure 9: Auction Time vs Maximum Bid

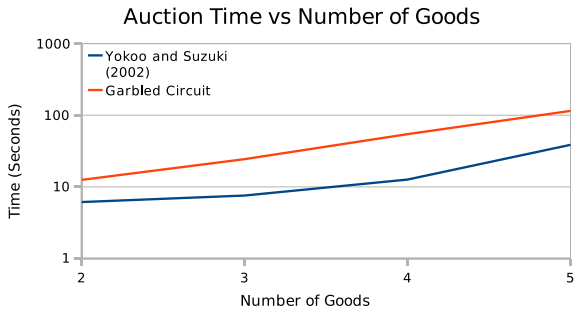


Figure 10: Auction Time vs No of Goods

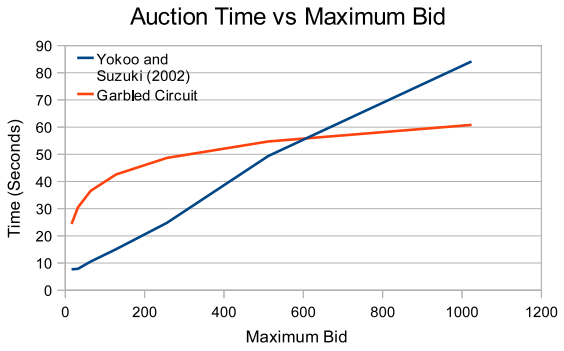


Figure 11: Auction Time vs Maximum Bid

son 2006) of the homomorphic auction protocol by Yokoo and Suzuki (Yokoo & Suzuki 2002). The time taken to complete the auction increasing exponentially as the number of goods increases. This is due to the number of possible allocations of goods increasing exponentially as the number of goods increases. For example, for 2 goods there are 4 possible allocations and for 3 goods there are 8 possible allocations. This is known as the combinatorial auction problem (CAP) which is NP complete and exponential. Depending on the particular auction taking place, there may be a large number of invalid allocations that can be removed to improve performance. Practical auctions can be done with fewer goods, for example a case study of industrial procurement auctions for cleaning contracts reported auctions with 9, 7, and 42 goods (Lunander & Lundberg 2009). As privacy preserving auctions would be particularly useful for high value goods, the running time and communication overhead of the auction would be less of an issue than for low value goods. Construction of a more compact combinatorial auction circuit would further reduce the auction overhead. The garbled circuit auction protocol performs worse than the protocol by Yokoo and Suzuki based on the number of goods. Pre-computation of the garbled circuit could provide a significant reduction in time taken to compute the auction.

Figure 11 compares the performance of garbled circuits with the performance of the homomorphic auction protocol with respect to the maximum bid. The garbled circuit auction protocol performs better than the homomorphic auction protocol when a large bid granularity is required due to a fundamental property of the bid vector representation used in the homomorphic auction protocol and other auction protocols that use a bid vector notation (Yokoo & Suzuki 2002, Suzuki & Yokoo 2002). When using a bid vector notation, increasing the bid vector size linearly increases the time taken to compute the auction and the maximum bid linearly. When using the garbled circuit auction protocol, increasing the number of bits in the price increases the time taken to compute the auction linearly but increases the maximum bid exponentially.

7 Conclusions

This paper has shown the development of an algorithm to construct a circuit composed of Boolean gates that can compute the result of a combinatorial auction. When combined with a privacy preserving auction protocol, based on general circuit evaluation, the algorithm can be used to conduct combinatorial auctions where only winning bids are made public. This is the first example of a combinatorial auction circuit to appear in the literature.

We have presented the concept of an auction circuit and described some of the building blocks we have used to create our algorithm. The algorithm to construct a combinatorial auction circuit is presented in detail. The size of the circuit created by our algorithm is presented. The size of the circuit grows linearly with the number of bidders. The size of the circuit increases exponentially with the number of goods as the number of possible combinations of goods in the auction also increases exponentially. The circuit size increases linearly as the number of available prices increases exponentially which provides an advantage for auctions where a large range of bids is required. We have shown that the communication overhead is feasible (6MB for an auction with 3 goods, a maximum price of 16, and 100 bidders). The garbled circuit auction protocol has also been shown to give comparable performance results to the homomorphic combinatorial auction protocol by Yokoo and Suzuki (Yokoo & Suzuki 2002). The garbled circuit auction protocol outperforms protocols that use a bid vector notation, such as the homomorphic auction protocol, when a large granularity of bids is required.

A Garbled Circuit Algorithms

This appendix describes our interpretation of and algorithms for the original single good garbled circuit auction protocol, and a simple worked example of a garbled circuit. This appendix explains ideas first presented in the original paper on the garbled circuit auction protocol (Naor et al. 1999), more details can also be found in the paper on the VPOT protocol (Juels & Szydlo 2003).

A.1 Table of Definitions

The following terms are used in the description of garbled circuits:

- Client: The entity that requests the auctioneer to conduct an auction.
- Auctioneer: Takes the details from the client and runs the auction. Communicates with the auction issuer to get the garbled circuit and garbled input values.
- Auction Issuer: Assists in running the auction. Should be from a separate organisation than the auctioneer. Garbles circuits and then assists the auctioneer in learning the garbled inputs.
- Bidder: Bids on items in the auctions.
- Auction Circuit: Circuit composed of Boolean gates that can be used to compute the result of an auction.

- Node: Boolean gate in an auction circuit.
- Wire: Link between two nodes of an auction circuit. A wire can have a value b of 0 or 1.
- W^0 and W^1: Multi-bit random values that are used to represent the 1 and 0 value of a wire.
- c: Result of a random permutation of a wires value b.
- $\langle W^b, c\rangle$: Garbled value of a wire. Formed by concatenating W for the value b of the wire with the result of the permutation c of the value b of the wire.
- g: The node function which calculates the output of the node based on the inputs. For example, for an AND gate $g(0,1) = 0$ and $g(1,1) = 1$.
- Gate Table: Each node in the auction circuit has a gate table that maps the garbled inputs to a garbled output.
- Output Mapping: Table that maps the garbled outputs to actual outputs. Each output wire has an output mapping.
- Pseudo Random Function $F(a,b)$: Pseudo random function F takes a as a seed and b as an argument and returns a random value. We use the SHA-1 hash function to represent this function.

A.2 Garbled Circuit Generation

To garble a circuit, the auction issuer executes the following algorithm on the nodes and wires of the auction circuit.

Algorithm 4
Procedure *GarbleCircuit*
Input: AuctionCircuit AC, RandomFunction F
Output: GateTable GT, OutputMapping OM
1. (∗ Assign random values to the wires ∗)
2. **for** wire $i \in$ AC
3. Randomly generate W_i^0 and W_i^1 corresponding to 0 and 1.
4. Choose a random permutation over {0,1}, $\pi_i : b_i \rightarrow c_i$.
5. (∗ Construct function tables for every node ∗)
6. **for** node $k \in$ AC with input nodes i,j
7. **for** $c_i \leftarrow 0$ **to** 1
8. **for** $c_j \leftarrow 0$ **to** 1
9. $GT(k)(c_i, c_j) \leftarrow$
10. $GetGTValue(i, j, k)$
11. (∗ Construct output mapping ∗)
12. **for** output wire $k \in$ AC
13. $OM(k, 0) \leftarrow \langle W_k^0, \pi_k(0)\rangle$
14. $OM(k, 1) \leftarrow \langle W_k^1, \pi_k(1)\rangle$

Algorithm 4 garbles an auction circuit. The first step is to assign random values to every wire of the auction circuit. Every wire has a value corresponding to 0 and 1 (W^0, W^1) assigned to it as well as a random mapping of its output π that maps the wires value b to c.

For every node in the auction circuit a table is constructed that, given the garbled input of the node, outputs the garbled output. If the node is an output node, an output mapping is also produced mapping the garbled output of the node to the actual output. These steps can only be performed with the knowledge of the random values assigned to all the wires. Algorithm 5 details the calculation done for an entry in the gate table. The tables for each node and the output mappings are then sent to the auctioneer to execute the circuit.

Algorithm 5
Procedure *GetGTValue*
Input: InputNode i, InputNode j, Node k
Output: bit [] Value
1. $Value \leftarrow$
2. $\{(W_k^{g(b_i,b_j)}, c_k) \oplus (F(W_i^{b_i}, c_j)) \oplus (F(W_j^{b_j}, c_i))\}$
3. return $Value$

A.3 Executing a Circuit

The following algorithm is executed by the auctioneer after it has received the GateTable and OutputMapping arrays from the auction issuer. The auctioneer will also have received the garbled inputs after completing the VPOT protocol with the bidders and auction issuer.

Algorithm 6
Procedure *ExecuteCircuit*
Input: AuctionCircuit AC, GateTables GT, OutputMapping OM, GarbledInputs GI, RandomFunction F
Output: ActualValues AV
1. (∗ Reset All Nodes ∗)
2. **for** Nodes $k \in$ AC
3. $Computed(k) \leftarrow$ **false**
4. (∗ Compute All Nodes ∗)
5. **repeat**
6. **for** Node k with input nodes i and j
7. **if** $((Computed(i) \cap Computed(j)) \cup (i \in GI \cap j \in GI)$
8. $GarbledOutput_k \leftarrow$
9. $GetGO(i, j, k, GT)$
10. $Computed(k) \leftarrow$ **true**
11. **until** All Nodes have been Computed
(∗ Convert Garbled Output to Actual Output ∗)
12. **for** output nodes o
13. **if** $(GarbledOutput_o = OM(o, 1))$
14. **then** $AV(o) \leftarrow 1$
15. **else** $AV(o) \leftarrow 0$

Algorithm 6 executes a garbled circuit given the auction circuit, gate tables, output mapping, garbled inputs, and random function. It loops through all the nodes in the auction circuit until they have all been computed. The gate tables are used to compute the garbled output of a node k with input wires i and j. Inputs i and j will have garbled input values of $\langle W_i^{b_i}, c_i\rangle$ and $\langle W_j^{b_j}, c_j\rangle$. From the garbled inputs the values c_i, c_j, $W_i^{b_i}$, and $W_j^{b_j}$ can be extracted from the concatenated garbled inputs. Then the garbled output can be computed using algorithm 7. Algorithm 7 uses the entry in the gate table for c_i and c_j as well as the output of the random function with seed $W_i^{b_i}$ and input c_j and with seed $W_j^{b_j}$ and input c_i. The output mapping is used to convert the garbled output to the actual output for an output node.

Algorithm 7
Procedure *GetGO*
Input: InputNode i, InputNode j, Node k, GateTables GT
Output: bit [] GarbledOutput
1. $GarbledOutput \leftarrow$
2. $F(W_i^{b_i}, c_j) \oplus F(W_j^{b_j}, c_i) \oplus GT[k](c_i, c_j)$
3. return $GarbledOutput$

A.4 A Simple Garbled Circuit

Figure 12 illustrates a small garbled circuit with an AND and an OR gate as well as the 'Random Values Assigned to Wires' which are the random values and

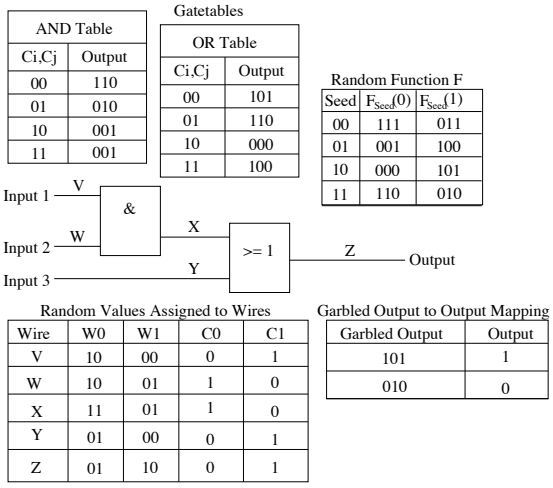


Figure 12: Garbled Circuit Example

permutation computed by the auction issuer and kept secret from any of the other parties taking part in the protocol. The auction issuer would have executed algorithm 4 to produce the random values assigned to wires, the gate tables and the garbled output to output mapping. The 'Random Function F' is available to any party in the protocol. The garbled value of a wire is set to $\langle W^b, c\rangle$ so for wire Z the garbled value of 0 is $\langle 01, 0\rangle = 010$.

To execute the circuit in Figure 12 the auctioneer would take the following steps:

- Find out the garbled input values. For say $V = 1$, $W = 1$, and $Y = 0$ the output should be 1. The garbled input value for V is 001, for W is 010, and for Y is 010. The garbled input value is the garbled value of the wire for the input value.
- Now we need to execute the gates. To execute the AND gate we use our garbled inputs and the gatetable. The output is $001 \oplus 111 \oplus 100 = 010$.
- Now we need to execute the OR gate. The output is $101 \oplus 001 \oplus 001 = 101$. Using the garbled output to output mapping we can see the output of the garbled circuit is 1.

This is a small example that shows how a garbled circuit works. A circuit that executes an auction has thousands of gates depending on the parameters of the circuit.

References

Baudron, O. & Stern, J. (2001), Non-interactive private auctions, *in* P. Syverson, ed., 'FC'01: Proceedings of the 5th Annual Conference on Financial Cryptography', Lecture Notes in Computer Science, Springer-Verlag.

Bubendorfer, K., Palmer, B. & Thomson, W. (2009), Dynamic ambient paradigms, *in* R. Buyya & K. Bubendorfer, eds, 'Market Oriented Grid and Utility Computing', Wiley, pp. 541–568.

Bubendorfer, K. & Thomson, W. (2006), Resource Management Using Untrusted Auctioneers in a Grid Economy, *in* 'proceedings of the Second IEEE International Conference on e-Science and Grid Computing (E-SCIENCE)', Amsterdam, Holland.

Cachin, C. (1999), Efficient private bidding and auctions with an oblivious third party, *in* 'CCS '99: Proceedings of the 6th ACM conference on Computer and communications security', ACM, New York, NY, USA, pp. 120–127.

Franklin, M. & Reiter, M. (1995), The design and implementation of a secure auction service, *in* 'Proceedings IEEE Symposium on Security and Privacy', IEEE Computer Society Press, Oakland, Ca, pp. 2–14.

Harkavy, M., Tygar, J. D. & Kikuchi, H. (1998), Electronic auctions with private bids, *in* 'WOEC'98: Proceedings of the 3rd conference on USENIX Workshop on Electronic Commerce', pp. 61–74.

Jakobsson, M. & Juels, A. (2000), Mix and match: Secure function evaluation via ciphertexts, *in* 'ASIACRYPT '00: Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security', Springer-Verlag, London, UK, pp. 162–177.

Juels, A. & Szydlo, M. (2003), A two-server, sealed-bid auction protocol, *in* 'FC '02: Proceedings of the 6th Annual Conference on Financial Cryptography', Springer-Verlag, pp. 72–86.

Kikuchi, H. (2001), (m+1)st-price auction protocol, *in* 'FC '01: Proceedings of the 5th International Conference on Financial Cryptography', Springer-Verlag, pp. 351–363.

Kurosawa, K. & Ogata, W. (2002), Bit-slice auction circuit, *in* 'ESORICS '02: Proceedings of the 7th European Symposium on Research in Computer Security', Springer-Verlag, London, UK, pp. 24–38.

Lipmaa, H., Asokan, N. & Niemi, V. (2002), Secure vickrey auctions without threshold trust, *in* 'FC'02: Proceedings of the 6th Annual Conference on Financial Cryptography', Springer-Verlag, pp. 85–101.

Lunander, A. & Lundberg, S. (2009), Do combinatorial procurement auctions lower cost? - an empirical analysis of public procurement of multiple contracts, Umeå Economic Studies 776, Umeå University, Department of Economics.

Naor, M., Pinkas, B. & Sumner, R. (1999), Privacy preserving auctions and mechanism design, *in* 'EC '99: Proceedings of the 1st ACM conference on Electronic commerce', ACM, pp. 129–139.

Peng, K., Boyd, C., Dawson, E. & Viswanathan, K. (2002), Robust, privacy protecting and publicly verifiable sealed-bid auction., *in* 'ICICS '02: Fourth International Conference on Information and Communications Security', pp. 147–159.

Perrig, A., Smith, S., Song, D. & Tygar, J. (2001), 'Sam: a flexible and secure auction architecture using trusted hardware', *Parallel and Distributed Processing Symposium., Proceedings 15th International* pp. 1764–1773.

Suzuki, K. & Yokoo, M. (2002), Secure combinatorial auctions by dynamic programming with polynomial secret sharing, *in* 'Sixth International Financial Cryptography Conference (FC-02)', Springer-Verlag, pp. 44–56.

Yao, A. C. (1982), Protocols for Secure Computations, *in* 'proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science', Chicago, IL, USA, pp. 160–164.

Yokoo, M. & Suzuki, K. (2002), Secure multi-agent dynamic programming based on homomorphic encryption and its application to combinatorial auctions, *in* 'proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)', ACM, New York, NY, USA, pp. 112–119.

Yokoo, M. & Suzuki, K. (2004), Secure generalized vickrey auction without thirdparty servers, *in* 'proceedings of the 8th International Financial Cryptography Conference (FC-2004)', Florida, USA.