Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2010) 1–10

Procedia Computer Science

International Conference on Computational Science, ICCS 2010

A protocol for verification of an auction without revealing bid values

Ben Palmer, Kris Bubendorfer, Ian Welch

School of Engineering and Computer Science
Victoria University of Wellington
PO Box 600, Wellington, New Zealand.
{ben,kris,ian}@ecs.vuw.ac.nz

Abstract

Auctions have been proposed for allocating computation resources for Cloud computing. However, many security issues exist with electronic auctions including insider trading, private information revelation, bid filtering, and auctioneers lying about auction results for their own profit. Privacy preserving auction protocols use cryptographic methods to keep losing bid values secret but many have no means of verifying their black box behaviour. This paper presents a protocol that allows participants to verify that an auction has run correctly without revealing bid values of other participants while increasing the robustness of the auction protocol.

Keywords:
Secure Auctions, Verification, Zero Knowledge

1. Introduction

Online auctions are a high profile method for consumer goods to be traded, for example consider the worldwide reach of eBay. However, its not just consumer goods that are being traded using online auctions. Recently, Amazon has followed the scientific community [1] in offering auctions as a means to allocate computational resources such as CPUs, memory and bandwidth [2] from the Elastic Computing Cloud. In the future, we expect that auctions will play an increasingly important role as a key way to allocate computational resources. However, there are two issues that must be addressed to make this vision a reality. The first is the use of an appropriate auction model for allocation of computation resources, and the second is addressing security concerns.

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 enables the auction to result in optimal allocations of goods to bidders.

Assuming the use of an appropriate auction model, the other main issue is security. Research into auction security has traditionally focused upon preserving the privacy of bidding information from as many parties as possible including the auctioneer. For example, the privacy preserving combinatorial auction protocol from Yokoo and Suzuki [3][1] hides the value of losing bids from auctioneers to prevent them from using this information to their advantage. However, existing protocols for privacy-preserving combinatorial auctions do not prevent other attacks such as misrepresentation of bids, removal or filtering of valid bids, or otherwise unfair manipulation of auctions.

This paper outlines how we have addressed these attacks by extending a privacy preserving combinatorial auction protocol to allow the conduct of the auction to be verified *while preserving the secrecy of bids*. This can be achieved by the use of zero knowledge proofs[2] to verify the actions taken by auctioneers while calculating the result of the auction. Furthermore, we have implemented this protocol and evaluated its efficiency and sensitivity to parameters such as encryption key length and number of goods.

2. Related Work

Naor, Pinkas, and Sumner developed a verifiable privacy preserving auction protocol using garbled circuits [4]. An auction issuer creates a garbled circuit that an auctioneer uses to conduct a privacy preserving auction. Privacy is preserved as long as the two parties do not collude. Bidders can verify their bids were counted and the auctioneer can use cut and choose verification to ensure the garbled circuit sent to it by the auction issuer is correct. Combinatorial auctions can be executed by the garbled auction protocol as long as there is a circuit able to compute the result of combinatorial auctions.

An auction scheme without third party servers that can perform a GVA has also been suggested by Yokoo and Suzuki [5]. They use the bidders to perform calculations based on the assumption that a bidder would have no incentive to be an active adversary against another bidders values. However, if a group of bidders is colluding this assumption is not true.

Lipmaa, Asokan, and Niemi have developed an auction scheme that partitions information between two parties so that one party working alone cannot subvert the auction [6]. A series of novel range proofs verify the auction, but the protocol does not support a GVA or combinatorial auction.

Kikuchi designed a verifiable single good auction protocol [7]. It makes use of verifiable secret sharing to allow bidders to verify auctioneer's calculations, and auctioneers to verify the bidder's encrypted bid values. However, it did not support combinatorial auctions and an extension by Yokoo and Suzuki [8] to accommodate combinatorial auctions did not include verification.

Finally, Brandt presented a bidder resolved single good auction protocol [9] where bidders use zero knowledge proofs to verify the auction result. The trust is distributed amongst the bidders, and is privacy preserving as long as at least one bidder is honest. This protocol has been extended to support combinatorial auctions [10], but the extension is not verifiable.

In summary, although some auction protocols use zero knowledge proofs to verify the auction result [9], we have found no auction protocol that is privacy preserving, verifiable, and able to compute the result of combinatorial auctions.

Zero Knowledge Proofs. Zero knowledge proofs (ZKPs) were first introduced by Goldwasser, Micali, and Rackoff [11] and are used to prove the validity of some assertion, without revealing any information other than the validity of the assertion.

Figure 1 illustrates a zero knowledge proof known as Ali Baba's cave. The prover Alice wants to convince the verifier Bob that she knows the secret password to open a locked door between R and S. To prove this while not revealing the secret password used, Alice and Bob conduct the following steps: (1) Bob waits at P while Alice goes to either R or S (commitment); (2) Bob goes to Q so that Alice may not move from (R) to (S) other than by the locked door (which she needs to know the secret to pass through) without him knowing; (3) Bob chooses either the top (R)

[1] For the remainder of this paper we refer to the auction protocol from Yokoo and Suzuki as the homomorphic auction protocol.

[2] Our verification protocol uses zero knowledge proofs of knowledge which we refer to throughout this paper as zero knowledge proofs or ZKP for brevity.

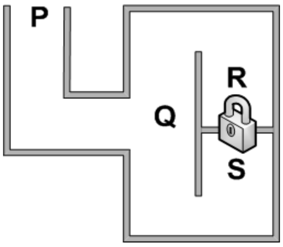


Figure 1: Ali Baba's Cave

or bottom (S) tunnel; (4) Bob challenges Alice to come out of the tunnel of his choice (challenge). Alice can only exit the correct tunnel 100% of the time if she knows the password; (5) If Alice does not know the secret words, there is a 50% chance she will come out from the wrong tunnel (response). Bob can then repeat steps 1-5 as many times as he wants to convince himself that Alice knows the secret word, but Bob will never learn the secret word himself.

3. The Homomorphic Auction Protocol

Central to our protocol is the existing homomorphic auction protocol. Figure 2 shows the main parties in the homomorphic auction protocol. There is a seller, bidders, and a group of auctioneers who execute the auction. The seller first arranges a group of auctioneers to compute the result of the auction. The seller then advertises the details of the auction to any interested bidders. Bidders then publish encrypted bids and the auctioneers calculate the result of the auction.

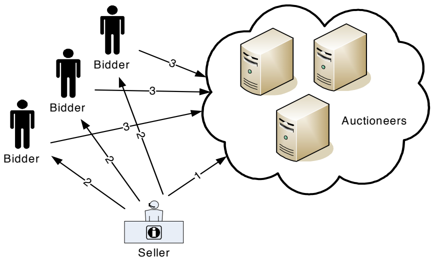


Figure 2: Parties in the Homomorphic Auction Protocol

The homomorphic auction protocol uses El-Gamal encryption [12] to provide privacy for the bids. El-Gamal is a homomorphic cryptosystem with indistinguishable encryption and randomisation. Multiplying two numbers encrypted using El-Gamal, when decrypted gives the same result as multiplying the unencrypted numbers. If we have two encryptions[3] of values M_1 and M_2 say $E(M_1)$ and $E(M_2)$ then $E(M_1)E(M_2) = E(M_1M_2)$. Indistinguishable encryption means that if you encrypt the same value twice using different arguments, it is computationally infeasible to tell they are the same number without access to the private key for decryption. We now describe the important features of the homomorphic auction protocol in relation to our verification protocol. More information on the homomorphic protocol can be found in Yokoo and Suzuki's paper [3].

3.1. Auction Graphs

Combinatorial auctions can be represented by auction graphs where nodes represent goods, edges between nodes represent the allocation of a subset of goods, and each complete path through the graph represents an allocation of the goods.

[3] We use the notation of E(M) to denote an encryption of the value M

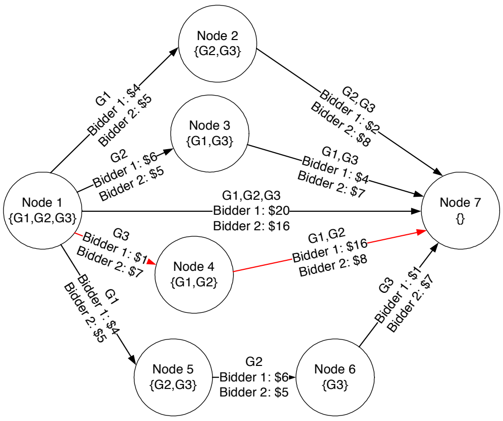


Figure 3: Example Auction Graph

Figure 3 provides an example auction graph for three goods $G1$, $G2$, $G3$ with two bidders. Each edge is labelled for the subset of goods it represents along with the bids of the two bidders for that allocation. A complete path through the graph is an allocation of goods and the auctioneer needs to find the optimal path through the graph to compute the auction. An optimal path is a complete path where there are no other paths that provide greater value. The optimal path is shown in this graph as the thinner line. The optimal allocation for this auction is to allocate $G3$ to bidder 2 for $7 and $G1, G2$ to bidder 1 for $16 for a total of $23. There may be more than one optimal path in a graph, but there should be no other paths that have a greater value.

3.2. *Bid Vectors*

Bids are represented in a bid vector form where an item in the bid vector is either an encryption of Z or 1. From the left, the number of Zs before the first 1 indicates the value of the bid vector so, for example, $(E(Z), E(Z), E(1), E(1), E(1))$ is a bid vector of length 5 with value 2.

3.3. *Randomising*

A bid vector can be randomised by multiplying the items in the bid vector by an encryption of 1. As the encryption used is homomorphic, the equation $E(M_1)E(M_2) = E(M_1M_2)$ holds and the multiplication of the items in the bid vector by an encryption of 1 does not change the value of the bid vector. If the Decisional Diffie-Hellman (DDH) problem is infeasible, one cannot determine whether a ciphertext is a randomised ciphertext of the original or not.

3.4. *Shift and Randomise*

An encrypted bid vector can have a constant c added to it without revealing the original encrypted value by using an operation called shift and randomise. Figure 4 shows the process for shifting and randomising a bid vector. First, the bid vector right by c and adding encryptions of Z to the spaces created. Then the whole bid vector is randomised by multiplying every item by indistinguishable encryptions of 1. Due to the randomisation, no information about the original bid vector value or the constant c can be obtained from the final bid vector. The maximum bid for the auction should be chosen to prevent overflow when adding a constant.

3.5. *Finding the Maximum Bid*

To find the maximum bid in a set of bid vectors, the bid vectors are multiplied together and decrypted from right to left to find the first item that does not decrypt to 1. Figure 5 shows an example where the maximum bid is Bid 3 with the value 4. As the decryption is performed from right to left and stops when the first Z value is found, no information about losing bids is revealed.

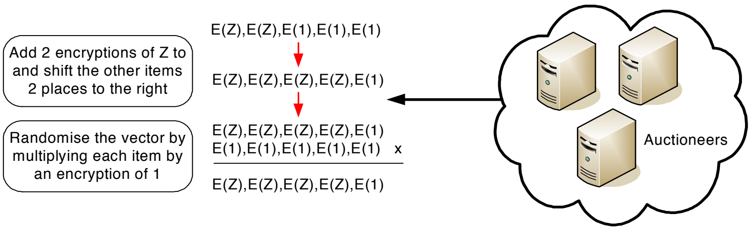

Add 2 encryptions of Z to and shift the other items 2 places to the right
Randomise the vector by multiplying each item by an encryption of 1
E(Z),E(Z),E(1),E(1),E(1)
E(Z),E(Z),E(Z),E(Z),E(1)
E(Z),E(Z),E(Z),E(Z),E(1)
E(1),E(1),E(1),E(1),E(1) x
E(Z),E(Z),E(Z),E(Z),E(1)
Auctioneers

Figure 4: Shift and Randomise

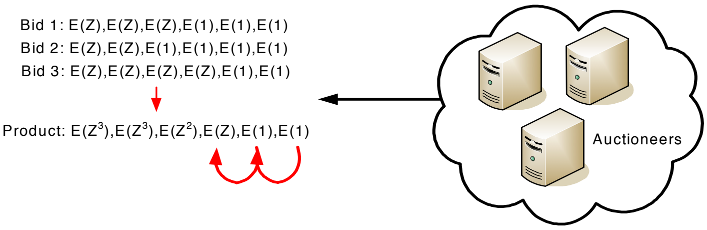

Bid 1: E(Z),E(Z),E(Z),E(1),E(1),E(1)
Bid 2: E(Z),E(Z),E(1),E(1),E(1),E(1)
Bid 3: E(Z),E(Z),E(Z),E(Z),E(1),E(1)
Product: E(Z^3),E(Z^3),E(Z^2),E(Z),E(1),E(1)
Auctioneers

Figure 5: finding the Maximum Bid

3.6. *Threshold Encryption*

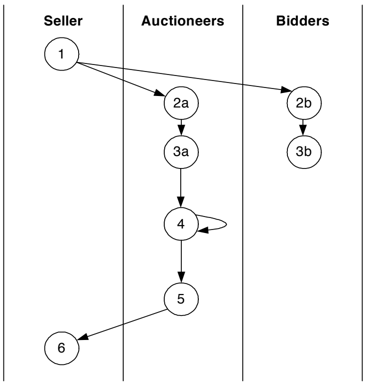
The homomorphic auction protocol can be executed with variable numbers of auctioneers. A single auctioneer can be used to execute the whole auction, but in this case if the auctioneer is malicious they can decrypt all the bids using their private key. A separate auctioneer can be allocated to each node in the auction graph with a unique private and public key. This means that a malicious auctioneer is only able to decrypt all the bids on the incoming edges of the node they are allocated to. A third option is to allocate a group of auctioneers to each node in the auction graph with a single public key and shared private keys using Shamir's secret sharing scheme [13]. In this case, unless the threshold value of auctioneers is corrupt, no information on the losing bids can be found out other than what needs to be known to execute the protocol. This paper uses a group of auctioneers allocated to each node in the graph using a (t, n) secret sharing scheme to allocate a unique private key to each auctioneer. At least t auctioneers are required to decrypt an encrypted item in a bid vector. When the auctioneers are required to decrypt an item, when computing the maximum bid for a link for example, at least t auctioneers are required to publish their shares of the decryption to the other auctioneers.

3.7. *The Auction Algorithm*

Figure 6 shows the entire homomorphic auction protocol. The first step is for the seller to advertise the details of the auction and arrange the group of auctioneers to execute the auction. In the second step, the auctioneers chosen to execute the auction publish the public keys for each link in the auction graph and the bidders who are interested in the auction register to bid. The third step involves the bidders creating and encrypting their bid vectors for each link in the auction graph and publishing these encrypted bids to the auctioneers. In the fourth step each group of auctioneers allocated to each node in the auction graph finds the maximum bid for their node (using the process in Section 5) in the graph and adds this value to each bid on each outgoing link from their node (using the process in Section 3.4).

4. Verification Protocol

We use two well known ZKPs to build our protocol: proof of equality of discrete logarithms [14] and proof that an encrypted item decrypts to one of two values [15]. We use the Fiat-Shamir heuristic [16] and the SHA512 hash function to make the proofs non-interactive ZKPs in the random oracle model. Non-interactive proofs allow auctioneers to publish the proofs once and removes the need to interact with other auctioneers that are verifying the



Step	Action
1	Advertise Auction, Arrange Auctioneers
2a	Publish Public Keys
2b	Register to Bid
3a	Wait For Bids
3b	Publish Encrypted Bid Vectors
4	Start from the Beginning Node in the Auction Graph and for every node: - Find the Maximum Bid for this Node - Use Shift and Randomise to add the Maximum Bid to the Bids on the Outgoing Links
5	Trace the Maximum Bids back through the Auction Graph to find the Optimal Path
6	Publish Winning Bidders and Prices

Figure 6: Auction Algorithm

result. A public bulletin board is also available that is readable by either auctioneers or bidders, but can only be written to by auctioneers.

At critical steps in the auction protocol, auctioneers publish a non-interactive ZKP to the other auctioneers that this has been done correctly. If one of the auctioneers finds that another auctioneers in the auction protocol has failed to provide a valid ZKP for a certain action, the auctioneer publishes that information to the bulletin board. A malicious auctioneer could claim that a correct auctioneer has behaved incorrectly when it has not. To prevent this, at least a threshold number of auctioneers is required to claim that an auctioneer has deviated from the auction protocol. This threshold value is the same value used in the threshold encryption of the homomorphic auction protocol as described in Section 3.6. Therefore losing bid values are kept secret and bidders can have confidence that the auction protocol executed correctly providing at least the threshold number of auctioneers are honest and indicate that the auction has executed correctly. In the case of at least the threshold number of auctioneers indicating that one of the actions taken by one of the auctioneers has failed the verification, the auction protocol can either continue ignoring any values returned by the offending auctioneer, or the auction result can be invalidated and the auction restarted.

Threat Model. The provers and verifiers in the verification protocol are assumed to be polynomially-bounded active adversaries that may try and prove incorrect assertions. It is assumed that any number of polynomially-bounded active adversaries may be colluding together to disrupt the verification protocol as long as there are less than the quorum or threshold value t used in the (t, n) threshold scheme. Different parties may collude, so a bidder and an auctioneer may collude together.

Verifiable Threshold El-Gamal Decryption. Threshold El-Gamal encryption ensures that a single auctioneer cannot decrypt all the bids for a link in the auction graph. Threshold El-Gamal uses a group of n auctioneers where the secret key x is shared using Shamir's secret sharing scheme $x \rightarrow (x_1, ..., x_n)$ [13] and requires the co-operation of at least t auctioneers to decrypt an encrypted value. Verifiable threshold El-Gamal uses ZKP of equality of logarithms to detect when an auctioneer publishes an incorrect decryption share with high probability. As long as less than t participants of the secret sharing scheme are malicious, the decryption can be carried out correctly even in the presence of malicious participants.

Verifying Valid Bid Vectors. Bids are represented in a bid vector form where an item in the bid vector is either an encryption of 1 or the publicly known value Z. From the left, the number of Zs before the first 1 indicates the value of the bid vector so, for example, $(E(Z), E(Z), E(1), E(1), E(1))$ is a bid vector of length 5 with value 2. A bid vector should contain no gaps where an encrypted 1 is between two encrypted Z values. An example of a bid vector with a gap is $bid_{gap} = E(Z), E(Z), E(1), E(Z), E(1)$.

Figure 7 shows the process used to verify that the bids are valid while not revealing their value. First the bidders submit their encrypted bids to the auctioneer in an alternate bid vector along with ZKPs that every item in the alternate bid vector is an encryption of a 1 or a Z [15]. Every item i in the alternate bid vector *alt* for a value v is calculated with the formula:

$$alt_i = \begin{cases} E(Z) & \text{if } i = v \\ E(1) & \text{otherwise} \end{cases}$$

The auctioneers then check the ZKPs that every item is an encryption of a 1 or a Z. The third step is for the auctioneers to decrypt the product of all the items in the bids vector. The auctioneers publish their shares of the decryption of the product along with ZKPs that their shares are correctly computed. The auctioneers then verify that the product of the items was correctly computed and that the product of the items decrypts to Z. Finally the alternate bid vector is integrated [17] to convert it to the standard form of the bid vector used in the homomorphic auction protocol. To integrate a bid vector one of the auctioneers computes the following function on the items of the alternate bid vector *alt* of length l starting with the item l and going down to item 1:

$$bid_i = \begin{cases} alt_i & \text{if } i = l \\ alt_i * bid_{i+1} & \text{otherwise} \end{cases}$$

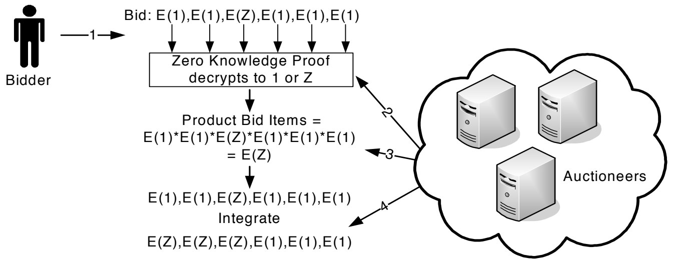

Bidder
1
Bid: E(1),E(1),E(Z),E(1),E(1),E(1)
Zero Knowledge Proof decrypts to 1 or Z
2
Product Bid Items = E(1)*E(1)*E(Z)*E(1)*E(1)*E(1) = E(Z)
3
E(1),E(1),E(Z),E(1),E(1),E(1)
Integrate
E(Z),E(Z),E(Z),E(1),E(1),E(1)
4
Auctioneers

Figure 7: Verifying Valid Bid Vectors

Verifying Maximum Bids. To calculate the maximum bid for a link of the auction graph, the product of all the incoming bid vectors is taken. This product vector is then decrypted from right to left to find the first value in the product vector that does not decrypt to 1. Figure 8 illustrates the process. Each auctioneer in the group publishes proofs that their share of the decryption has been correctly calculated using a proof of equality of discrete logarithms.

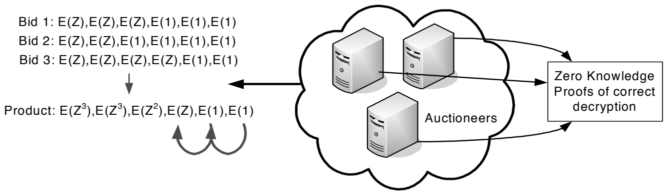

Bid 1: E(Z),E(Z),E(Z),E(1),E(1),E(1)
Bid 2: E(Z),E(Z),E(1),E(1),E(1),E(1)
Bid 3: E(Z),E(Z),E(Z),E(Z),E(1),E(1)
Product: $E(Z^3),E(Z^3),E(Z^2),E(Z),E(1),E(1)$
Auctioneers
Zero Knowledge Proofs of correct decryption

Figure 8: Verifying Maximum Bid

Verifying Shift and Randomise. To prove in zero knowledge that an auctioneer has correctly added a constant m and re-randomised a bid vector of length l the auctioneer first calculates m encryptions of Z and publishes ZKPs of equality of logarithms to prove that these m items are encryptions of Z to the other auctioneers. The auctioneer then shifts the new bid vector right by m places, adds the m encryptions of Z, and publishes this shifted bid vector to the other auctioneers. The auctioneer calculates l encryptions of 1 and publishes ZKPs of equality of logarithms to prove that these l items are encryptions of 1 to the other auctioneers. The auctioneer then publishes the shifted and randomised bid vector by multiplying the items in the shifted vector by the l encryptions of 1. Figure 9 provides an example of this process where the auctioneer is adding the constant $m = 2$ to a bid vector of length $l = 5$.

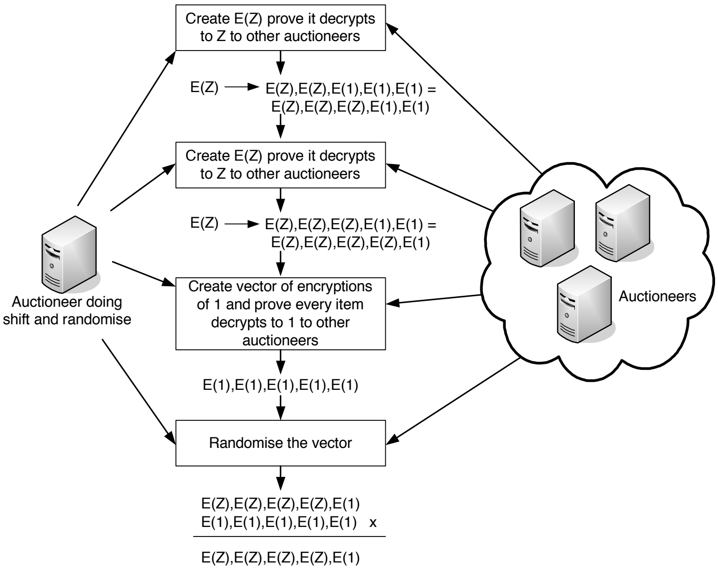

Create E(Z) prove it decrypts to Z to other auctioneers
E(Z)
E(Z),E(Z),E(1),E(1),E(1) = E(Z),E(Z),E(Z),E(1),E(1)
Create E(Z) prove it decrypts to Z to other auctioneers
E(Z)
E(Z),E(Z),E(Z),E(1),E(1) = E(Z),E(Z),E(Z),E(Z),E(1)
Create vector of encryptions of 1 and prove every item decrypts to 1 to other auctioneers
E(1),E(1),E(1),E(1),E(1)
Randomise the vector
E(Z),E(Z),E(Z),E(Z),E(1)
E(1),E(1),E(1),E(1),E(1) x
E(Z),E(Z),E(Z),E(Z),E(1)
Auctioneer doing shift and randomise
Auctioneers

Figure 9: Auctioneer adding 2 to the bid vector E(Z),E(Z),E(1),E(1),E(1)

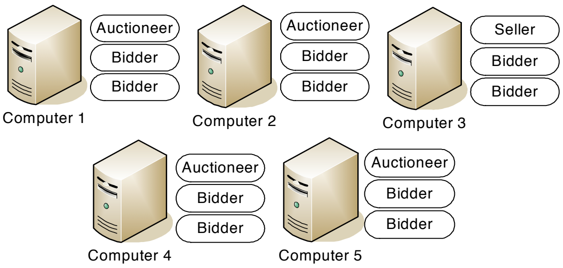


Figure 10: Homomorphic Auction and Verification Protocol Test Setup

5. Algorithmic Complexity

Table 1 presents the upper bound of the complexity of the original homomorphic auction protocol and the verifiable extension based on the number of modular exponentiations. The verification protocol adds an overhead of $2^g(9bl + 2ln + 2bn) + 2bgn$ exponentiations. While this is a significant overhead, the main influence on the number of modular exponentiations is still the number of goods in the auction.

Homomorphic Protocol	Verifiable Protocol
$2^g(4bl + ln) + bgn$	$2^g(13bl + 3ln + 2bn) + 3bgn$

Table 1: Complexity. g is no of goods, b is no of bidders, l maximum bid for a link

6. Experimental Results

The verification and auction protocols were implemented in Java. Figure 10 shows the testing setup. The auctions were run on a group of five computers. Each computer hosts from 2 to n bidders. One computer hosts the seller and the other four host the auctioneers. The first three auctioneers to respond to the sellers request to host an auction will

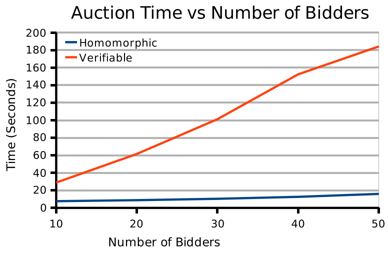


Figure 11: No. Bidders vs Auction Time

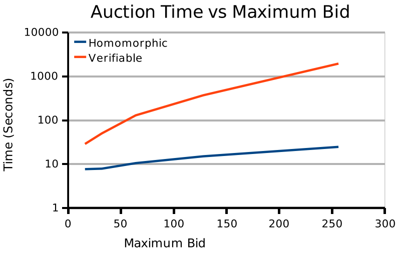


Figure 12: Maximum Bid vs Auction Time

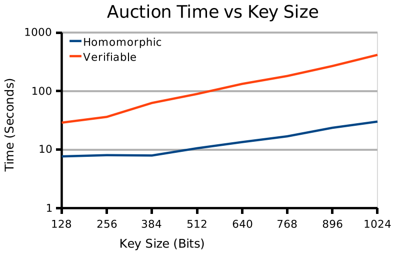


Figure 13: Key Size vs Auction Time

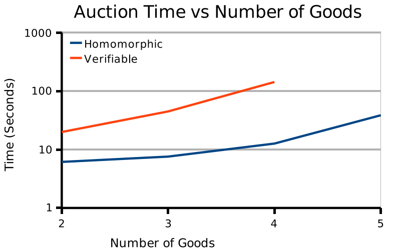


Figure 14: No. Goods vs Auction Time

run the auction protocol for this auction. A subsequent auction may utilise a different partition of the auctioneers. The bid values were chosen to show the upper bounds of the protocol. Due to the fact that the auctioneers decrypt the bid vectors from right to left when working out the maximum bid, the bids were chosen to be randomly either one or two. This means that the auctioneers will have to do more work than if the bids were higher values. In our tests we used a $(2, 3)$ threshold scheme for the encryption. The default values for the parameters were ten bidders, a maximum price of sixteen, three goods, and a key size of 128. All tests record the total auction time. This includes the time taken by bidders to encrypt and submit bids as well as the time taken by the auctioneers to calculate the result of the auction, publish ZKPs that their actions are correct, and verify the proofs of the other auctioneers.

Figure 11 shows a linear increase in the number of bidders causing linear growth in the time taken to compute the auction. The verifiable auction protocol adds a significant overhead which is due mainly to the extra proofs that must be generated by every bidder to prove that their bid is valid.

Figure 12 shows a linear increase in the maximum bid resulting in an exponential growth of the time taken to compute the auction result. This is true for both the homomorphic auction protocol and the verifiable auction protocol.

Figure 13 shows the time taken to compute the auction increasing exponentially as the key size is increased linearly. This is due to the extra computation requirements of operating on the larger encrypted values. The homomorphic auction protocol does not seem to increase when using a key size from 128 bits to 384 bits and only seems to experience exponential growth for key sizes greater than 384. This may be due to a minimum overhead being created by the communication costs of the protocol that influence the time taken to compute the auction more than the computational requirements of computing with larger numbers up to a key size of 384. This is also reflected in the verifiable auction protocol where the exponential growth seems to happen for key sizes greater than 256. As there is more computation done on the larger numbers in the verifiable auction protocol it would have an effect on the auction time for smaller key sizes than the homomorphic auction protocol.

Figure 14 shows the effect of increasing the number of goods on the auction time. Both the homomorphic auction protocol and the verifiable auction protocol have exponential growth in auction time as goods are increased. Increasing the number of goods in a combinatorial auction results in an exponential increase in the number of possible allocations. This is known as the combinatorial auction problem (CAP) which is NP complete and exponential. Although a straight line in this graph would be expected as the goods increase, the extra increase could be added due to some inefficiency in the implementation such as an inefficient graph creation algorithm that results in extra time taken for each good added. There is no result for the verifiable auction protocol for five goods, as the Java version used was restricted to a maximum of 734MB of memory per process and the auctioneers for the verifiable auction protocol require more memory.

We have stated that our motivation for using a privacy preserving verifiable auction protocol is to allocate computation resources in a cloud computing environment. While the results of these experiments show that the time taken to complete a verifiable auction would add a large overhead to the resource allocation if done on a task by task basis, we envisage these auctions being used in a multi-level scheduling architecture. A light weight scheduler such as Falkon [18] would be used to allocate tasks while combinatorial auctions are used by a provisioner to reserve large amounts of resources. In this scenario the provisioner would be taking part in high value auctions where the verification of the auction result provides confidence for resource providers that the auction result has been computed correctly.

It is interesting to note that the experimental results we have taken agree with the algorithmic complexity. The verification protocol does add a significant overhead to the homomorphic auction protocol.

7. Conclusion

We have added verification to an existing privacy preserving combinatorial auction protocol. Provided less than a threshold of auctioneers are malicious, losing bid values are kept secret and auction participants can have confidence in the auction result. With the addition of verification, the auction protocol is a privacy preserving, verifiable, combinatorial auction protocol. While the verification protocol adds an overhead to the auction protocol, it increases confidence in the auction result and can improve the robustness of the protocol by detecting and removing invalid bids or malicious auctioneers. While the algorithm is NP complete, experimental results have shown useful auctions with three goods, fifty bidders, 128 bit key size, and sixteen possible bid values are computed in approximately three minutes. While this may be too expensive for resource allocation for individual tasks, we can envisage a verifiable privacy preserving combinatorial auction protocol being used by a provisioner to reserve large blocks of resources to be allocated to tasks by a light weight scheduler.

[1] R. Buyya, K. Bubendorfer (Eds.), Market-Oriented and Utility Computing, John Wiley, 2009.

[2] T. Claburn, Amazon auctions cloud computation, Info. Week.

[3] M. Yokoo, K. Suzuki, Secure multi-agent dynamic programming based on homomorphic encryption and its application to combinatorial auctions, in: 1st Int. Conference on Autonomous Agents and Multiagent Systems, ACM, New York, NY, USA, 2002, pp. 112–119.

[4] M. Naor, B. Pinkas, R. Sumner, Privacy preserving auctions and mechanism design, in: 1st ACM conference on Electronic commerce, ACM, 1999, pp. 129–139.

[5] M. Yokoo, K. Suzuki, Secure generalized vickrey auction without thirdparty servers, in: 8th Int. Financial Cryptography Conference, Florida, USA, 2004.

[6] H. Lipmaa, N. Asokan, V. Niemi, Secure vickrey auctions without threshold trust, in: 6th Annual Conference on Financial Cryptography, Springer-Verlag, 2002, pp. 85–101.

[7] H. Kikuchi, (m+1)st-price auction protocol, in: 5th Int. Conference on Financial Cryptography, Springer-Verlag, 2001, pp. 351–363.

[8] K. Suzuki, M. Yokoo, Secure combinatorial auctions by dynamic programming with polynomial secret sharing, in: Sixth Int. Financial Cryptography Conference, Springer-Verlag, 2002, pp. 44–56.

[9] F. Brandt, How to obtain full privacy in auctions, Int. J. of Information Security 5 (4) (2006) 201–216.

[10] J. D. Nzouonta, An algorithm for clearing combinatorial markets, Master's thesis, Florida Insitute of Technology (2003).

[11] S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof systems, SIAM J. of Computing 18 (1) (1989) 186–208.

[12] T. E. Gamal, A public key cryptosystem and a signature scheme based on discrete logarithms, in: Advances in cryptology, Springer-Verlag New York, Inc., New York, NY, USA, 1985, pp. 10–18.

[13] A. Shamir, How to share a secret, Commun. ACM 22 (11) (1979) 612–613.

[14] D. Chaum, T. P. Pedersen, Wallet databases with observers, in: 12th Annual Int. Cryptology Conference on Advances in Cryptology, Springer-Verlag, London, UK, 1993, pp. 89–105.

[15] R. Cramer, R. Gennaro, B. Schoenmakers, A secure and optimally efficient multi-authority election scheme, in: Workshop on the theory and application of cryptographic techniques on Advances in cryptology, Vol. 1233, Springer-Verlag, London, UK, 1997, pp. 103–118.

[16] A. Fiat, A. Shamir, How to prove yourself: Practical solutions to identification and signature problems, in: 6th Annual Int. Cryptology Conference on Advances in Cryptology, Springer-Verlag, London, UK, 1987, pp. 186–194.

[17] M. Abe, K. Suzuki, M+1-st price auction using homomorphic encryption, in: PKC '02: Proceedings of the 5th Int. Workshop on Practice and Theory in Public Key Cryptosystems, Springer-Verlag, London, UK, 2002, pp. 115–124.

[18] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde, Falkon: a fast and light-weight task execution framework, in: SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing, ACM, New York, NY, USA, 2007, pp. 1–12. doi:http://doi.acm.org/10.1145/1362622.1362680.