Resource Management Using Untrusted Auctioneers in a Grid Economy

Kris Bubendorfer and Wayne Thomson
School of Mathematics, Statistics and Computer Science,
Victoria University of Wellington, New Zealand.
kris@mcs.vuw.ac.nz, wayne.thomson@mcs.vuw.ac.nz

Abstract

The CORA (Coallocative, Oversubscribing Resource Allocation) architecture is an auction based resource reservation system that makes combinatorial allocations of resources to clients. The focus of this paper is on the use of cryptographic tools in CORA to remove the need for trust in the resource auctioneer. One of the nice properties of this approach is that the auctioneers can be drawn from an arbitrary pool of untrusted peers, without the need to establish pre-existing trust or restrict the role of auctioneer to a trusted system service. This approach results in more flexibility in the design of large economic systems, with the potential for wide distribution of load amongst many auctioneers. In addition, only the winners of the auction and the prices they pay are revealed while all other bid values are kept secret. It is our belief that future growth or commercialisation of large scale Grid systems requires the provision of such mechanisms to share the wide pool of Grid brokered resources such as computers, software, licences and peripherals amongst many users and organisations. This paper encapsulates an overview of our design, our experiences of implementing two different secure auction protocols and the performances that we have achieved.

1 Introduction

One of the vital components of any Grid computing infrastructure is the resource broker. A Grid resource broker is effectively an arbiter for access to a Grid's computational resources and therefore its performance and functionality has a wide ranging influence on the utilisation and performance of the Grid. As the size of a Grid network increases, the resource broker becomes a choke point and ultimately limits the performance of, or access to, the full computational power of the Grid. Indeed if one were to envision a truly global Grid, then such an enterprise must potentially permit any arbitrary combination of resources and users to interact. Market based mechanisms, such as auctions, have often [3, 6, 7] been promoted as a solution for scalable resource economies because they are naturally decentralised, efficient and produce optimal allocations. Another advantage of such market based mechanisms is that they are a natural fit with the UCP (Utility Computing Providers) [10, 17] scenario and efforts towards Grid commercialisation [9, 13]. When the resources being allocated are spread across varied administrative domains and ownership, the allocations of the resource allocator (in our case auctioneer) must provide confidence in the outcome, yet disclose as little private valuation information as possible - especially in a competitive economic environment. Thus the role of the auctioneer must be carefully scrutinised.

In general, existing Grid resource allocations and associated trust models are based on the knowledge that people have about each other and their organisations. This model suits the scientific establishment well, where there are a small number of large well known organisations, individually possessing large collections of computing resources. However, this model is ultimately limited in scalability by its reliance on the *friend of a friend* chain for access to resources. There are also no standard mechanisms to deal with intentional or unintentional misuse of the Grid.

In the CORA (Coallocative, Oversubscribing Resource Allocation) [1, 4] project, we are investigating an economic management model based upon Vickrey auctions for resource reservation. The primary focus of the CORA architecture is to achieve higher resource utilisation [2] through controlled oversubscription of resources to auctions than is normally achieved in conventional resource auctions. Central to user acceptance of a market based resource reservation system such as CORA is belief in its security. The security requirements for CORA go beyond the usual need for confidentiality, availability, integrity and access control by including a requirement for trustworthy auctions. A trustworthy auction process is central to the proper allocation of resources. For example, what happens if the auctioneer chooses the winner because the auctioneer's integrity has been subverted, or the auctioneer uses bids to collect sensitive information for a competing bidder. The simplest

but least flexible approach is to simply trust nominated auctioneers, but this blackbox approach restricts who can host an auction thereby limiting the scalability of the underlying Grid system.

Rather than nominating a trusted auctioneer, in CORA we are applying cryptographic algorithms to establish trustworthy auctions that preserve privacy and to detect malfeasance. One of the significant advantages of this approach is that the auctioneers can be drawn from an arbitrary pool of untrusted peers, without the need to establish pre-existing trust or restrict the role of auctioneer to a trusted system service. Thus trust is no longer seated in the resource brokers, but is derived from the cryptographic confidence embodied in the trustworthy auction algorithms. While there is considerable overhead introduced by the cryptographic processing, this overall approach results in more flexibility in the design of large systems – with the potential for wide distribution of resource broking load amongst many peers.

Many of the lessons we have learnt in developing CORA are applicable to the wider Grid community. In this paper we firstly give an overview of trust in electronic auctions, secondly we present the two cryptographic techniques (based on homomorphic encryption [30] and an adaption of Shamir's polynomials [28]) that we have implemented, and lastly we provide our performance measurements for the two techniques.

2 Background

The focus of this paper is on the use of cryptographic tools in CORA to remove the need for trust in the resource auctioneer. However, the background as to why we need to establish this trust is also important. Auctions are an efficient solution to the challenge of distributed resource allocation in both economic [3, 6, 7] and non-economic [21] resource allocation systems. Many computational auctions adopt the 2^{nd} price Vickrey auction protocol, i.e., the winner pays the price established by the second highest bid. This protocol is popular as the 2^{nd} price mechanism results in truthful bids [32], where the bid valuations reflect the true worth to the bidder. It also has properties such as low messaging overhead and efficiency of allocations.

When it comes to computational auctions the problem is that most existing resource auction systems [6] focus on a single representative resource. Indeed, it may not be possible to achieve QoS goals with single representative resource as the basis for resource allocations. While the results of single resource auctions are easy to compute, this representation is not realistic as execution resources form an indivisible set, related and conditional upon the availability of each other. Piecewise negotiation for individual resources will not often provide a usable result, let alone optimal allocation. Game theorists term this as the combinatorial allocation problem (CAP), in which a set of components have a synergistic value that exceeds the sum of the individual parts. Because of synergistic combinations and possible substitutions, bidders have preferences not just for particular resources, but for collections of resources.

The Generalised Vickrey Auction (GVA) [20] extends the original Vickrey auction protocol to address the CAP. Solving a single GVA auction is *NP*-hard [26], and for this reason there are a number of optimised variations [23, 24], and approximations [14, 18] that reduce the computation time. As there is no 'second-price' in a combinatorial auction the price paid by the winner is their bid less a discount. The discount is calculated by removing the winner from the auction and recomputing the result. The difference between the two values is the winner's discount. This discount mechanism is equivalent to the 2^{nd} price mechanism in a non-combinatorial auction and therefore retains the desirable dominant strategy of truthful bidding in private value (when a good is for consumption rather than resale) auctions, that is, bidding your true value will always give the best return regardless of other bidders strategies.

Besides the CAP, there are other problems that need to be faced when using auction protocols for distributed resource allocations. In particular *all* of the known auction protocols have limitations in practice, and an exhaustive analysis of these protocol considerations is detailed in [27]. In the Vickrey auction for instance, a compromised auctioneer can undetectably issue false bids to inflate the value of the second bid or reduce the winner's discount. Likewise, the values of past bids can be collected and either used in future auctions, or passed on to colluding bidders – *"Even if current information can be safeguarded, records of past behaviour can be extremely valuable, since historical data can be used to estimate willingness to pay."* [31]. The cryptographic auction algorithms in this paper focus on the Vickrey auction protocol and solves the problem of the lying auctioneer and the potential for a corrupt auctioneer to reveal the bid values. This is achieved by hiding the information from the auctioneers in such a way that the winners and the prices can still be determined.

3 Establishing Trust in Auctions

Until recently there was little recourse but to design auction based allocation systems with an auctioneer as a trusted service. However, this approach tends to centralise designs and lacks openness, transparency and verifiability. Recently there have been significant research efforts to determine if an auctioneer is acting in a trustworthy manner or to even remove the need for the auctioneer to be trusted at all. Essentially trust in an auction can be established in one of four ways:

1. Pre-existing trust (i.e., system components)

2. Reputation services (i.e., perceived trust)
3. Bid-encryption schemes (i.e., procedural trust)
4. Threshold schemes (i.e., distributed trust)

Reputation services rate the performance of an auctioneer based on reports from the participants after the auction [8]. Trust can be delegated [15] to bootstrap new auctioneers into the system. However, the problem of initial trust remains, as does the problem of verification – how do participants verify the auction process without revealing potentially sensitive valuation information? For example, zero knowledge proofs can be used to allow the auctioneer to prove that it correctly calculated the winner without revealing bid values [19].

Bid encryption schemes dispense with the need to initially trust an auctioneer, indeed, the issue of whether an auctioneer is trustworthy is no longer relevant. Here, cryptographic protocols are used that make it impossible (or excessively expensive) for an auctioneer to learn anything useful from, or to manipulate the outcome of, an auction. For example, Noar [22] proposed a general method for computing any auction protocol securely including combinatorial auctions using a method known as garbled circuits.

Threshold schemes are based upon secret sharing [28]. These schemes allow trust to be placed in a set of auctioneers rather than a single auctioneer. As long as a certain number (a quorum) of auctioneers are not corrupt and execute the auction protocol correctly, a minority of malicious auctioneers cannot subvert the protocol and manipulate the auction. For example, the bids could be encrypted using public key cryptography and require cooperation of multiple auctioneers to decrypt the bids so the winner can be computed [12]. The requirement for cooperation of a minimum number of correct auctioneers prevents the auction being manipulated and prevents the auctioneers from learning bid values during execution of the protocol.

4 A Trustworthy Auctioneer

We have implemented two different threshold bid-encryption schemes for CORA. The first is based on the secure 1st price homomorphic auction scheme by Yokoo and Suzuki [33] which is used to encrypt the bids while allowing their use in winner determination. Distributed decryption is used to prevent malicious auctioneers learning any bids [25]. Our initial experiences working with this protocol were reported in [5]. This scheme was later revised by Suzuki and Yokoo [30] as a 2^{nd} price auction by computing the discount in a matrix. This implementation is not very efficient and we have since extended this 2^{nd} price scheme to compute the discount more efficiently utilising the dynamic programming technique from [33]. The implementation has also been extended to determine the prices as well as the winners. The results that we are presenting in this paper include our new efficient discount computation, and show that the subsequent performance improvement is considerable.

The second scheme is based on a modification of Shamir's [28] secret sharing polynomials using the degree of the polynomial rather than the constant to encode the secret [16]. This scheme is then extended in [29] for 1st price combinatorial auctions. The key to both schemes is in the representation of the bid values. As we have published our early work on the homomorphic scheme in [5], we will dedicate more explanation in this paper to the polynomial scheme.

4.1 Homomorphic Representation

The SGVA protocol represents values as vectors composed of elements encrypted using a homomorphic cryptographic scheme, and defines operations that allow addition and comparison without revealing the values themselves. Each element in the vector is either the encryption of the value one or a common public value chosen by the auctioneers. The value encoded in the vector is equal to the number of encrypted common public values. Note that each element may represent a unit larger than 1. For example, an auction with a vector of size 10 could have bids \$1 to \$10 or \$10 to \$100 using a \$1 or \$10 unit value respectively. A bidder's *weight collection* is the set of all its vectors for each possible combination of goods in the auction.

Addition of two vectors depends upon the use of a homomorphic cryptographic scheme that allows addition of encrypted values without needing to decrypt the values. Our implementation uses the Elgamal public key encryption scheme [11] and allows two vectors to be added together by componentwise multiplication of their vector elements. Besides adding two vectors representing bid values we also need to add constants. An efficient approach is to left shift vector components as many times as the value of the constant. Addition of a random constant is used to hide individual bid values while allowing auctioneers to calculate the maximum bid value from the collection of bids.

Winner computation requires comparing bid values, but comparison of two vectors cannot be done directly because, due to randomisability, two vectors representing the same value will contain different component values. Therefore the SGVA protocol performs comparison as a two-step process. In step 1, all the values to be compared are multiplied together to find the largest. In step 2, the result is decrypted one element at a time from left to right until we find an element equal to one. The position of this element (or properly the element one to its left) is the greatest price from the collection of vectors. This reveals the maximum value without revealing individual values.

4.2 Polynomial Representation

As was stated earlier, the polynomial representation encodes the bid values in the degree of a polynomial. Unfortunately a good description of the polynomial 1st price auction is lacking in the literature, hence we include here our interpretation of the scheme. The protocol is described in detail in the following five steps.

Step 1: An auction initiator publishes an initial graph of the auction and each auctioneer publishes a unique resolving value. The initial auction graph contains a node for each combination of goods with edges between nodes which can be bid on. A constant value c is published for weight resolution and a threshold value t is published that is used to prevent interference from less than t corrupt auctioneers.

Example: Ten evaluators $\{e_1, e_2, \ldots e_9, e_{10}\}$ publish resolve values $\{1, 2, 3, \ldots 8, 10\}$. The auction initiator publishes an auction graph containing the three available combinations ($\{g_1\}, \{g_2\}, \{g_1, g_2\}$) of 2 goods (Fig. 1).

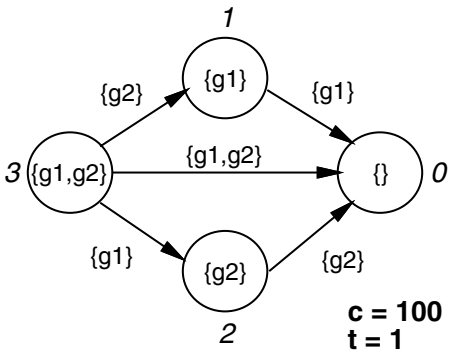

Figure 1. An initial graph for two resources.

Step 2: Bidders calculate their valuations by iterating through each edge in the published graph. The valuation is added to a threshold modifier $t(|j - i|)$ to form the final weight w. A bid is created for each auctioneer by generating a random polynomial of degree w and constant 0 and solved using the auctioneer's resolve value. The solved value s is then sent with the node identifiers to the auctioneer. As each auctioneer receives bids, a new edge is created on their local copy of the graph from j to i with weight s.

Example: Evaluations and bid weights for two bidders b_0 and b_1 are given in table 1. The polynomials used in this example have coefficients of one, which compromises security but makes the examples simpler. b_0 generates a set of polynomials corresponding to the weights including $x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x$ and $x^2 + x$ for edges $(3, 0)$ and $(1, 0)$ respectively. The resolved polynomials sent to evaluators for both bidders are shown in table 2.

Step 3: Each evaluator computes their share of the optimal value of the auction, which is the sum of bidder valuations on the greatest path through their graph. Two properties of polynomials make this possible [29]: the degree of the sum of two polynomials is equal to the larger of the two and the degree of the multiplicative sum of two polynomials is equal to the sum of the degrees. Therefore the cost of a path is discovered by multiplying its bids together, and the greatest cost path is found by adding alternative paths together.

Edge	Goods	b_0		b_1	
		valuation	w	valuation	w
(n_3, n_2)	$\{g_1\}$	1	2	1	2
(n_1, n_0)	$\{g_1\}$	1	2	1	2
(n_3, n_1)	$\{g_2\}$	2	4	1	3
(n_2, n_0)	$\{g_2\}$	2	4	1	3
(n_3, n_0)	$\{g_1, g_2\}$	4	7	2	5

Table 1. Bidder evaluations for two resources.

Evaluator	$(3,2)$	$(3,1)$	$(2,0)$	$(1,0)$	$(3,0)$
b_0					
e_1	2	4	4	2	7
e_2	6	30	30	6	254
...	...	...	...	...	...
e_9	90	7380	7380	90	5380839
e_{10}	110	11110	11110	110	11111110
b_1					
e_1	2	3	3	2	5
e_2	6	14	14	6	62
...	...	...	...	...	...
e_9	90	819	819	90	66429
e_{10}	110	1110	1110	110	111110

Table 2. Bids created by b_0 and b_1.

The most efficient way of calculating a share is to iteratively calculate the cost of each node from root node 0. The node cost $f(x)$ is the sum of the cost of all alternative paths to node 0 from node x. The cost of a path from node x through $x - 1$ where the nodes are connected by at least one edge is $f(x - 1)$ multiplied by the sum of the cost of all alternative edges between the nodes. Note that $f(0)$ is always set as 1.

Example: Evaluator e_1 calculates it's share of the node costs.

$$f(1) = 2 + 2 = 4$$
$$f(2) = 4 + 3 = 7$$
$$f(3) = ((4 + 3) * 4) + ((2 + 2) * 7) + (7 + 5) = 68$$

Node costs for all evaluators are given in table 3.

Step 4: Binary search is used to discover the optimal value. As the bids are distributed amongst the evaluators,

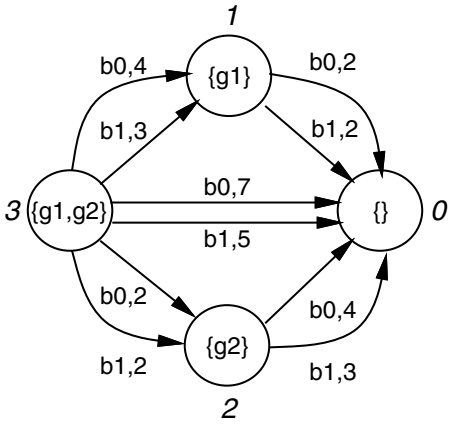

Figure 2. Graph maintained by e_1 after bids placed.

Evaluator	f(1)	f(2)	f(3)
e_1	4	7	68
e_2	12	44	1372
...	...	...	...
e_9	180	8199	8398908
e_{10}	220	12220	16599020

Table 3. Node cost for all evaluators.

valuations must be reconstructed by interpolating the shares published by evaluators. Using Lagrange interpolation a polynomial can be recovered with a degree equal to the optimal value o. However the degree of the polynomial is one less than the number of shares used, so it is the number of shares to publish which needs to be discovered. A slightly modified version of binary search is used to find the correct number of shares. The initial smallest possible optimal value u is 0 and the maximum optimal value m is $E-1$ which are both used to determine the initial search value. A recursive binary search calculates the current search value $d = \lfloor (m-u)/2 \rfloor + u$ and requests $d+1$ evaluators publish masked shares. Masks are created by M masking agents who generate a random polynomial of degree $d+1$ and constant c. If the Lagrange polynomial using $d+1$ shares has a constant equal to $M * c$, then $o \leq d$ and m becomes $d-1$. If the constant is not equal to $m * c$ the $o > d$ and u becomes $d+1$. The result is $o = d$ where $o \leq d$ and $o > d-1$.

Example: The minimum optimal value is set at 0, the maximum at 9 and so initially $d = 4$. Masks are sent to five evaluators from the mask publishers. The masks are polynomials of degree 4 and coefficient 1 (for simplicity) which are resolved with the respective evaluator's resolve value. Each of the ten available mask publishers generate the masking polynomial $x^4 + x^3 + x^2 + x$ and send masks $104, 130, 220, 440, 880$ to e_1, e_2, e_3, e_4, e_5. The evaluators add the total masking value to their share of the optimal value and publish it. The published shares are:

$$\begin{array}{ll} e_1 & (1, 1108) \\ e_2 & (2, 2672) \\ e_3 & (3, 13474) \\ e_4 & (4, 61528) \\ e_5 & (5, 222560) \end{array}$$

The Lagrange polynomial from these five points is $1988x^4 - 15211x^3 + 46185 - 60334x^2 + 28480$. As the constant is not equal to 100, $o > 4$. Table 4 shows the binary search for the optimal value with $o = 7$.

d	Constant
4	28480
7	1000
5	−17720
6	6040

Table 4. Using binary search to find the optimal value.

Step 5: Once the optimal value is discovered an optimal path can be traced to discover an optimal allocation of goods. There can be multiple optimal paths (optimal allocations), but for the purpose of this scheme it does not matter which optimal allocation is used. As the cost of an optimal node n_x is the greatest path cost from n_x to the root node, the cost of an optimal edge added to the cost $f(y)$ of connected node y is $f(x)$. Starting with $x = N-1$, and $f(N-1) = o$, an optimal edge is found by iteratively evaluating the cost of each edge from x added to the cost of the destination node. Binary search as in step four is used to check edges from x for $f(x)$. Edge costs will differ from the actual cost and are modified accordingly by subtracting the threshold modifier.

Example: The edges from n_3 are searched and all but the edge by b_0 on (n_3, n_0) are not equal to $f(3)$. Once this edge is found the search ends because the destination is n_0. The actual cost is found as $7 - (1 * (3-0)) = 4$ and the optimal allocation is published as b_0 wins $\{g_1, g_2\}$ for \$4.

5 Polynomial Evaluation

In this section, we explore the performance of the polynomial algorithm to determine the sensitivity of the algorithm to the four main variables: the number of bidders, the threshold parameter, the number of resources being auctioned, and the permitted bid range. The selection of these

values has an impact on the tractability of the polynomial algorithm. Each of the performance results was measured on a commodity Dell with a 128MB JVM, Java 1.5, 3GHz Pentium 4 with a 1Gbit Ethernet. Other than the variable under test, the parameters selected for the performance measurements were: ten bidders, three items, max bid of 5, threshold set to 1 and a total of 8 graph nodes.

5.1 The Number of Bidders

The number of bidders in an auction is one variable that is outside the control of the auction. However, it is worth noting that as a minimum, a Vickrey auction needs at least four bidders to provide an optimal allocation[32].

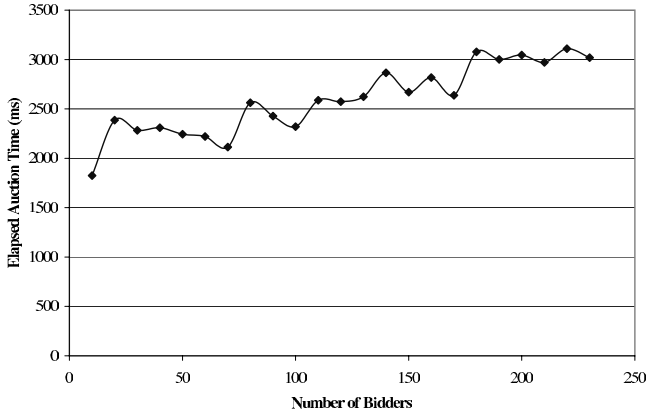

Figure 3. An increase in the number of bidders has a linear effect on elapsed auction time.

Figure 3 shows that as the number of bidders increases, the elapsed auction time experiences linear growth. As each bidder is added the only impact is that additional edges are added to the auctioneers' graphs, the effect is a small linear time increase.

5.2 The Threshold Parameter

The threshold t controls the tolerance of the scheme to corrupt auctioneers. As the threshold is increased the impact on the elapsed auction time is exponential, as shown in figure 4. This is a direct result of storing the secret in the degree of the polynomial and therefore increasing the threshold increases the degree of the polynomial. This increases the time to perform the Lagrange interpolation due to the increase in number of shares required to solve polynomials of increasing degree. Consequently, one of the problems with this approach is that any increase in t results in a larger number of auctioneers, which is an undesirable cyclic dependency.

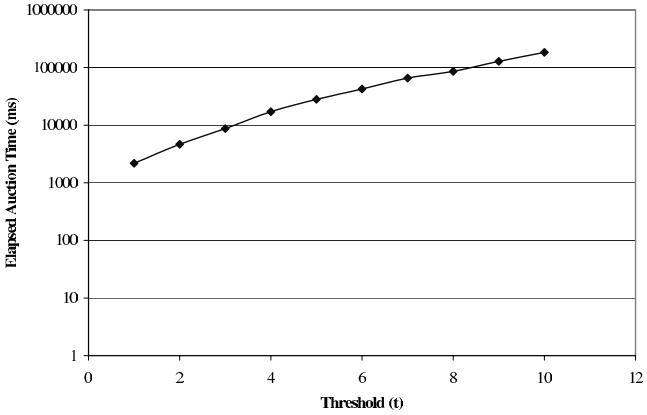

Figure 4. The effect of increasing threshold (t) on elapsed auction time.

5.3 Maximum Bid

Increasing the size of the maximum bid allows for finer grained bids to be made, otherwise we must scale the bidders' valuations within a limited range. A larger bid results in a polynomial of higher degree. This has similar performance implications to increasing the threshold and for the same reasons. As shown in figure 5 the effect on elapsed auction time is exponential.

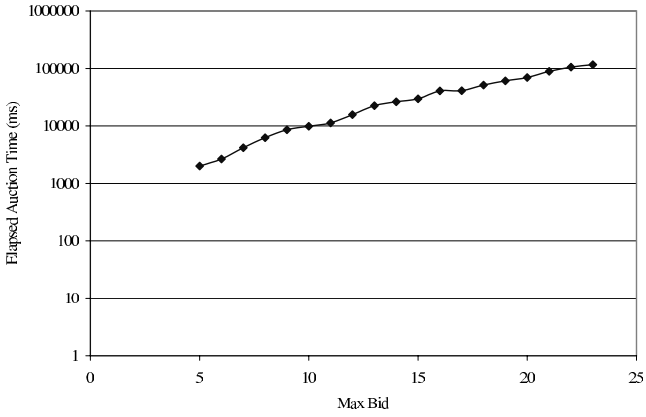

Figure 5. The effect of increasing maximum bid on elapsed auction time.

5.4 Combinations of Goods

The number of available goods for sale in a combinatorial auction is usually the major performance bottleneck. An auction with two items means there are three possible combinations that need to be computed and evaluated, adding a third item increases the number of combinations to seven. This growth of combinations is $f_n = 2^n - 1$, and solving such a combinatorial problems is always *NP*-hard.

Figure 6 compares the overall results of all three of our implementations. As can be seen, the polynomial implementation does not perform as well as either the 1st or 2^{nd} price homomorphic implementations. It is also worth pointing out that the 2^{nd} price homomorphic auction performs better with 5 or more resources than the 1st price homomorphic auction - due to our most recent dynamic programming implementation.

The significance of these results is that they show that we can compute the combinatorial 2^{nd} price allocations of up to five goods within a reasonable time – 3.4 seconds for 5 goods or 11 seconds for 6 goods. This gives us a working baseline on which to apply approximations to the CAP, that we intend to incorporate into future iterations of our system.

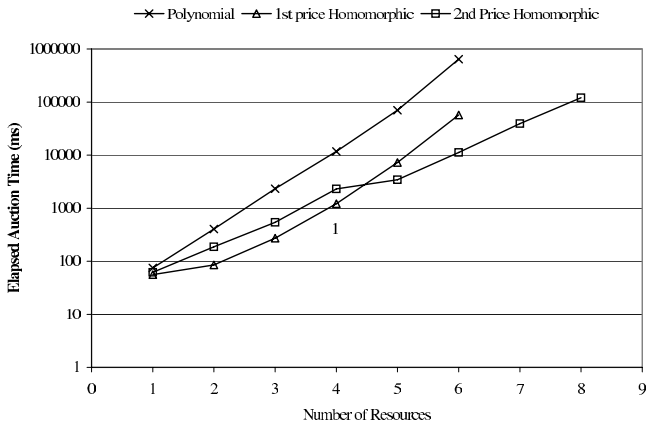

Figure 6. The effect of increasing the number of resources on winner determination time

6 Conclusions and Future Work

The essential ideal behind all trustworthy auction protocols is to cryptographically hide part of the bid information from the auctioneers, while still permitting the auction protocol to determine the winners and their combinatorial allocations. This approach removes the need for pre-existing trust and enables an auction to be held by any adhoc group of auctioneers who are not individually trusted. One of the advantages of adopting trustworthy auction protocols in CORA and other Grid resource management systems is to provide the potential for scalability through a wide distribution of auction load amongst many peer auctioneers. However, these cryptographic approaches are expensive and when used in combination with combinatorial auctions limit the number of resources to approximately seven. Nonetheless, this is a significant improvement over the use of a single representative resource for scheduling decisions.

The results in this paper have focused on the performance of the polynomial scheme – while we are able to extend the polynomial scheme to compute the 2^{nd} price discount, its 1st price performance indicates that this is not worthwhile. It has also become clear that the polynomial scheme has a number of disadvantages when compared to the homomorphic schemes. In particular, the number of auctioneers must increase as the security or the size of the bids are increased. It is this property that is significantly detrimental to its overall performance. However, it is clear that a trustworthy auction scheme can be implemented within a reasonable performance envelope using the 2^{nd} price homomorphic scheme.

As part of our future work we intend to; utilise approximations to help reduce the overhead incurred when computing the combinatorial allocations; develop a means for bidders to verify that the auction has produced the expected results; develop a hybrid feedback system so that only well behaved and well performing auctioners populate the auctioneer pool; and implement and experiment with other cryptographic approaches such as garbled circuits.

7 Acknowledgments

This work exists as part of a larger research effort, and as such includes indirect contributions from a number of people. In particular Ian Welch has co-supervised research students and has co-authored related papers.

References

[1] K. Bubendorfer. Improving Resource Utilisation in Market Oriented Grid Management and Scheduling. In *To appear in proceedings of the 4th Australasian Symposium on Grid Computing and e-Research (AusGrid 2006)*, volume 48, Hobart, Tasmania, Australia, January 2006.

[2] K. Bubendorfer. Improving resource utilisation in market oriented grid management and scheduling. In *Proceedings of the 4th Australasian Symposium on Grid Computing and e-Research*, January 2006.

[3] K. Bubendorfer and J. H. Hine. Resource Discovery and Negotiation in the NOMAD System. In *in Proceedings of ACSC2005, The Twenty Eigth Australasian Computer Science Conference*, volume 27, Newcastle, NSW, Australia, January 2005.

[4] K. Bubendorfer, P. Komisarczuk, K. Chard, and A. Desai. Fine Grained Resource Reservation and Management in Grid Economies. In *Proceedings of the 2005 International Conference on Grid Computing and Applications*, pages 31–38, Las Vegas, Nevada, USA., June 2005.

[5] K. Bubendorfer, I. Welch, and B. Chard. Trustworthy Auctions for Grid-style Economies. In *In the proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid (CCGrid06)*, volume 1, pages 386–390, Singapore, May 2006. IEEE.

[6] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in

Grid computing. *Concurrency and Computation: Practice and Experience*, 14:1507–1542, 2002.

[7] C.-H. Chien, P. H.-M. Chang, and V.-W. Soo. Market-Oriented Multiple Resource Scheduling in Grid Computing Environments. In *Proceedings of Advanced Information Networking and Applications (AINA'05)*, volume 1, pages 867–872, Taipei,, March 2005.

[8] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 207–216, New York, NY, USA, 2002. ACM Press.

[9] T. Dimitrakos, D. M. Randal, F. Yuan, M. Gaeta, G. Laria, P. Ritrovato, B. Serhan, S. Wesner, and K. Wulf. An Emerging Architecture Enabling Grid Based Application Service Provision. In *Seventh International Enterprise Distributed Object Computing Conference (EDOC'03)*, pages 240–251, Brisbane, Queensland, Australia, September 2003.

[10] P. Eerola, B. Konya, O. Smirnova, T. Ekelof, M. Ellert, J. R. Hansen, J. L. Neilsen, A. Waananen, A. Konstantantinov, and F. Ould-Saada. Building a Production Grid in Scandinavia. *IEEE Internet Computing*, 7(4):27–35, 2003.

[11] T. Elgamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):69–472, July 1985.

[12] M. Franklin and M. Reiter. The Design and Implementation of a Secure Auction Service. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 2–14, Oakland, California, USA., 1995. IEEE Computer Society Press.

[13] S. Graupner, V. Kotov, A. Andrzejak, and H. Trinks. Service-Centric Globally Distributed Computing. *IEEE Internet Computing*, 7(4):36–43, 2003.

[14] M. M. Halldórsson. Approximations of Weighted Independent Set and Hereditary Subset Problems. In *Computing and Combinatorics, 5th Annual International Conference, COCOON 99, Lecture Notes in Computer Science 1627 Springer*, pages 261–270, 1999.

[15] L. Kagal, S. Cost, H. Chen, T. Finin, and Y. Peng. An Infrastructure for Distributed Trust Management. In *Workshop on Norms and Institutions in Multiagent Systems, Autonomous Agents*, Montreal, Canada., may 2001.

[16] H. Kikuchi. (M + 1)st-Price auction protocol. In *FC2001: Proceedings of the Fifth International Conference on Financial Cryptography*, volume LNCS 2339, pages 351–363, Grand Cayman, February 2001.

[17] P. Komisarczuk, K. Bubendorfer, and K. Chard. Enabling virtual organisations in mobile networks. In *IEE 3G2004 Conference*, pages 123–127, London, UK, October 2004.

[18] D. Lehmann, L. I. Oćallaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.

[19] H. Lipmaa, N. Asokan, and V. Niemi. Secure vickrey auctions without threshold trust. In M. Blaze, editor, *Financial Cryptography*, volume 2357 of *Lecture Notes in Computer Science*, pages 87–101. Springer, 2002.

[20] J. K. MacKie-Mason and H. R. Varian. Generalized Vickrey Auctions. Working paper, University of Michigan, 1994.

[21] T. W. Malone, R. E. Fikes, K. R. Grant, and M. T. Howard. Enterprise: A Market-like Task Scheduler for Distributed Computing Environments. In H. B.A, editor, *The Ecology of Computation*, pages 177–205. Elsevier Science Publishers (North-Holland), 1988.

[22] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *EC '99: Proceedings of the 1st ACM conference on Electronic commerce*, pages 129–139, New York, NY, USA, 1999. ACM Press.

[23] N. Nisan and A. Ronen. Computationally feasible VCG mechanisms. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 242–252, New York, NY, USA, 2000. ACM Press.

[24] D. C. Parkes. An Iterative Generalized Vickrey Auction: Strategy-Proofness without Complete Revelation. In *Proceedings of the AAAI Spring Symposium on Game Theoretic and Decision Theoretic Agents*, CA, USA, Mar. 2001.

[25] K. Peng, C. Boyd, E. Dawson, and K. Viswanathan. Five sealed-bid auction models. In *CRPITS '03: Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003*, pages 77–86, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[26] M. H. Rothkopf, A. Pekeč, and R. M. Harstad. Computationally Manageable Combinatorial Auctions. Technical Report 95-09, DIMACS, Center for Discrete Mathematics and Theoretical Computer Science, Rutgers, USA, Apr. 1995.

[27] T. Sandholm. Limitations of Vickrey Auction in Computational Multiagent Systems. In *In Proceedings of the Second International Conference on Multiagent Systems (ICMAS-96), Kyoto, Japan*, pages 299–306, December 1996.

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

[29] K. Suzuki and M. Yokoo. Secure Combinatorial Auctions by Dynamic Programming with Polynomial Secret Sharing. In *FC2002: Proceedings of the Sixth International Conference on Financial Cryptography*, volume LNCS 2357, pages 44–56, Southampton, Bermuda, March 2002.

[30] K. Suzuki and M. Yokoo. Secure generalized vickery auction using homomorphic encryption. In *Financial Cryptography, 7th International Conference, FC 2003*, volume 2742 of *Lecture Notes in Computer Science*. Springer, 2003.

[31] H. R. Varian. Economic Mechanism Design for Computerized Agents. In *Proceedings of Usenix Workshop on Electronic Commerce*, July 1995.

[32] W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[33] M. Yokoo and K. Suzuki. Secure Multi-agent Dynamic Programming based on Homomorphic Encryption and its Application to Combinatorial Auctions. In *Proceedings of the first joint International Conference on Autonomous Agents and Multiagent Systems*, Bologna, Italy, July 2002. ACM.