Chapter 24

Trust in Grid Resource Auctions

Kris Bubendorfer, Ben Palmer and Wayne Thomson

24.1 Introduction

Trust is a concept that we humans implicitly understand, but we have difficulty in applying our 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 (referral). However, trust can be misplaced, and the degree of risk underestimated. Essentially, just because something is trusted, this does not mean it is also trustworthy.

Consider the collapse of the 1995 Barings Bank [1] in which Nick Leeson a floor manager for Barings Trading abused his position of trust by hiding losses made in the futures and options markets over three years. The mechanism he used was cross trades made to a special error account that was not visible to auditors in the main London office. Leeson was also head of settlement operations for his unit, which short-circuited normal accounting and auditing safeguards. As in most cases the employee had a satisfactory history within the company before being placed in this position of trust. Without a perfect oracle it is probably not possible to make better trust evaluations. What was

really missing in the Barings Bank collapse were additional mechanisms (checks and balances) to ensure that its employees could not act in an unauthorized way.

We draw an analogy here to resource allocation mechanisms. Even if a community broker or auctioneer is trusted we still cannot have confidence in its trustworthiness without additional mechanisms to audit and constrain its actions. Within the scope of a single organization this may not be an issue, however when we cross organizational boundaries and potentially include the potential for profit or loss, then we need better mechanisms than traditional *blind trust* for a Market Oriented Grid.

24.1.1 The Need for Trustworthy Resource Auctions

Imagine that a number of independent resource providers (let's call them Bob and Jane) have surplus resources and wish to sell the rights to use these resources via Alice, their friendly local 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. The auction house itself is hosted on 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.

There are a number of potential problems with this auction. Alice can freely examine the bids from Bob and Jane. She can then reveal this information to others giving them a competitive advantage, or she can simply collect bid information over time to manipulate

the market by setting reserves and so on. The bid information as such, represents commercially sensitive information from the resource providers and this information should ideally be kept private. Even if Alice is willing to keep the bid information private, Sam could extract the information directly from memory or if it were encrypted, extract the key from memory. Alice might also favor Jane over Bob, by discarding all or a portion of Bob's bids – at the expense of Bob and the clients. Likewise Sam could filter bids preventing Alice from including them in the auction. If Alice or Sam were also a resource provider, then the incentive to cheat is considerable. "Even if current information can be safeguarded, records of past behavior can be extremely valuable, since historical data can be used to estimate the willingness to pay" – Hal Varian [2].

The first step in solving these problems is to ensure that bids are kept private, that is, hidden from Alice or Sam. At first it seems that this is impossible, as Alice would be unable to compute the winner of the auction. However, we can exploit certain cryptographic techniques that allow Alice to compute the outcome of the auction, without revealing anything other than the winner and the final price. Such schemes are known as a *privacy preserving* scheme, whereby Alice (or Sam) cannot reveal or use the bid information to manipulate the market. In addition, in a 2nd price or Vickrey auction [3], Alice cannot inflate the prices to obtain a higher commission – a problem known as the corrupt auctioneer. We will detail these techniques later in this chapter.

The second step is to ensure that the auction protocol has been executed correctly and that all bids have been considered to preclude the possibility of filtering or favoritism.

This can be achieved using a *verification* scheme – some of which permit any party in the auction to verify offline that the auction was held correctly.

Essentially we have three main requirements for the holding trustworthy resource auctions in a multi-organizational (or open) Grid:

1. We should be able to avoid having to trust a single auctioneer;
2. We should be able to provide strong bid privacy; and,
3. We should be able to verify that the auction was conducted correctly.

24.2 Secure Auction Protocols for the Grid

Removing the need to establish trust enables Grid allocation architectures that are user-centric, peer oriented, open and dynamic. From this flexibility we should see improvements in reliability, availability and accessibility. Resource auctions can be executed safely using *any* computing resources contributed by *any* provider. As such, as the size of the Grid increases, additional untrustworthy computing resources can be deployed or redeployed dynamically to meet any subsequent growth in the number of resource auctions.

Table 42.1 presents a taxonomy of auction protocols that feature some form of cryptographic security. The first column shows whether the auction protocol is verifiable by anyone (public) or whether it is verifiable only by specific participants of the protocol (group). The next column shows whether the auction protocol supports

combinatorial auctions. The bid privacy trust model is shown in the next column and indicates if the auction protocol uses a single trusted auctioneer, two party trust, or a multi party trust scheme. Finally the table shows the bid privacy provided by the protocols using the following privacy levels (adapted from [4]):

- Level 0: Only the winning bidder/s and the price they paid are revealed.
- Level 1: In addition to the information revealed by level 0, one other piece of information is revealed.
- Level s: In addition to the information revealed by level 0, it is also possible to recover bid statistics. For example, the average bid and the standard deviation of bids.
- Level *: All of the bids are revealed to the auctioneer after the auction closes.

Auction Scheme	Verifiability		Combinatorial Auction Support	Bid Privacy Trust Model			Bid Privacy Level			
	Grp	Pub		Single Trusted Auctioneer	2 Party Trust	Multi Party Trust	0	1	s	*
Franklin & Reiter [5]	•					•				•
HKT [6]						•	•			
2-Server [7]	•				•			•		
Garbled Circuits [8]	•		•		•		•			
Non Interactive [9]				•			•			
Polynomial [10]	•					•	•			
No Threshold Trust [5]	•				•				•	
Polynomial GVA [11]			•			•	•			
Homomorphic [12]			•			•		•		
Extended HKT [13]		•				•	•			
Five Models [14]		•				•			•	
SGVA [4]			•			•	•			
Yet Another [15]		•				•	•			
Receipt Free [16]		•			•		•			
Combinatorial Bidder Resolved [17]			•			•	•			
GW Micali [18]		•				•	•			

Verifiable Scheme [19]		•		•						•
Bidder Resolved [20]		•				•	•			
Extended Verifiable Homomorphic [21]		•	•			•	•			

Table 42.1: Privacy Preserving Auction Taxonomy

Consider the three requirements outlined in section 24.1.1:

1. We should be able to avoid having to trust a single auctioneer:

   Schemes that use a single trusted auctioneer include [9] and [19] are not in themselves trustworthy protocols. Some other schemes use two party trust [8], [7], [5] and [16] where bids are kept private unless there is collusion between these two parties. All remaining schemes use some form of multiparty trust, where the operation is spread over many independent parties normally implemented using multi party trust. Undermining this form of auction would require collusion between a significant number of corrupt parties. A two party trust scheme has the practical advantage that it is often easier to find two servers from separate organizations than it is to find a large group of independent servers as is used in a multi party trust scheme. On the other hand, multi party trust can provide increased reliability, availability, and accessibility.

2. We should be able to provide strong bid privacy:

   In the schemes [5] and [19] a trusted Auctioneer learns bids after the bid opening phase of the auction. This requires a large amount of trust be placed in the Auctioneer to not sell or otherwise leak bid information. Rather than releasing specific bids, the schemes [5] and [14] leak bid statistics while the scheme [7] leaks a partial ordering of bids. To provide the requirement of strong bid privacy

we should look at using one of the schemes with the highest level of bid privacy (level 0).

3. We should be able to verify that the auction was conducted correctly:

   Apart from providing confidence in the result of auctions, verifiability can increase the robustness of the solution by allowing participants to check intermediate results. For example, an auction evaluator conducting threshold decryption can prove that their share of the decryption is correctly calculated. Most verifiable auctions were initially group verification, where *involved* parties could verify the outcome of the auction. A more recent development has been public verifiability, where *any* entity can verify the result of the auction. Public verification can also be added to existing schemes [13] and [21] through the use of techniques such as zero knowledge proofs.

24.3 Auction Protocol Case Studies

As can be gathered from reading the taxonomy in section 24.2, there are many secure auction protocols of varying capability that can be used as the basis for a trustworthy grid auction. In this section we present three of the most Grid suitable protocols as worked examples.

24.3.1 How to represent combinatorial auctions

Combinatorial auctions have been represented in a number of ways including graphs, matrices, and circuits. Two of the auction protocols presented in this chapter represent auctions using graphs, while the third uses a circuit.

A graph can be used to solve combinatorial auctions where nodes represent unallocated goods, edges between nodes represent the allocation of a subset of goods to bidders and each complete path represents an allocation of the goods. Figure 24.1 provides an example auction graph for three goods *{G1, G2, G3}* with two bidders *B1* and *B2*. Edge allocations are shown under edges, bids are presented above edges and nodes are labeled n_0 to n_{N-1} where N is the number of nodes in the graph. The highest path cost from n_5 to n_i is represented by it's f-value $f(i)$. The highest bid for edge (n_5, n_2) is $4 and as node n_5 has no incoming edges $f(5) = 0$ and so $f(2) = 4$. The other f-values are $f(3) = 9$, $f(4) = 1$, and $f(1) = 10$. To determine $f(0)$, each of the incoming edges are compared, and (n_3, n_0) provides the highest cost path with $(n_3, n_0) + f(1) = 19$. Highest bids are recorded at each step and the winning allocation is *B2* wins *G2* for nine dollars and *B1* wins *G1* and *G3* for ten dollars.

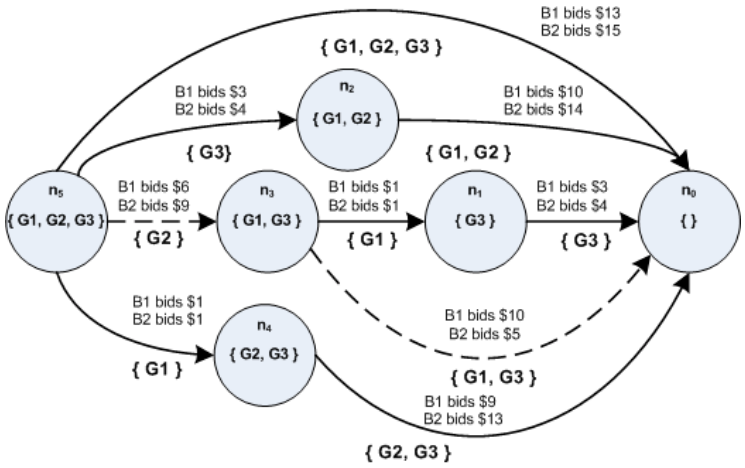

Figure 24.1 An auction graph for three goods G1, G2 and G3. The optimal path is indicated with the dashed line.

An auction circuit can also be used to represent a combinatorial auction and is composed of a series of Boolean gates. The circuit has a set of inputs, the gates of the circuit, and a set of outputs.

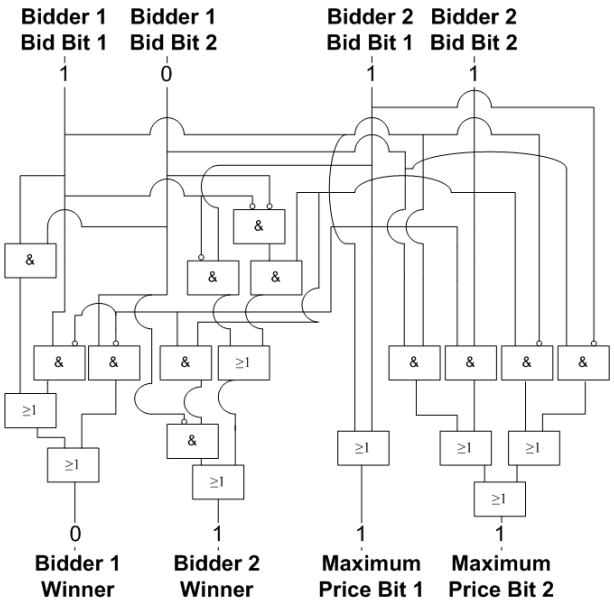

Figure 24.2: A Simple Auction Circuit

Figure 24.2 shows a Boolean circuit that is being used for a trivial auction with one good, two bidders and two bits representing the price. The circuit has inputs of the bids for the two bidders represented by two bits. The outputs are composed of two Boolean values that indicate whether bidder 1 was the winner or if bidder 2 was the winner. The other two outputs indicate the bits of the maximum or winning price. The example in Figure 24.2 shows Bidder 2 winning the auction with a maximum price of 11. To further illustrate consider an further example where bidder 1's input is 10 and bidder 2's input is 01, the output is Bidder 1 Winner = 1, Bidder 2 Winner = 0, Maximum Price Bit 1 = 1, and Maximum Price Bit 2 = 0.

24.3.1 Garbled Circuits

The principle idea of a garbled circuit is to act as a replacement for the 'trusted party' in transactions between mutually distrustful parties. Garbled circuits were presented in Yao's seminal 1986 paper [22] 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. In essence, a garbled circuit acts as a black box, which will evaluate an input without revealing anything but the output for that one value. A garbled circuit involves the creation of a set of Boolean gates to compute a function, and then the garbling of this circuit to obfuscate the input and intermediate values but still allow execution of the function.

To achieve privacy on the input and intermediate values of the circuit, the circuit creator assigns random values and a permutation to wires in the circuit keeping these hidden from the party that will execute the circuit. If these values are known to the circuit executor then the privacy of inputs is lost.

To garble a circuit, the circuit creator assigns every wire connecting the gates a random value, W^1, corresponding to the *1* value of the wire and a random value, W^0, corresponding to the *0* value of the wire. Every wire is also assigned a random permutation over *{0, 1}* π: $b \rightarrow c$ from the actual value of the wire *b* to the permutated value *c*. The garbled value of a wire with an actual value of *b* is defined as $<W^b,c>$ where the random value associated with b is concatcenated with the result of the permutation of *b.* Figure 24.3 shows a wire with an actual ungarbled value of *b*. The random values and permutation tables are shown above the wire and the garbled values below.

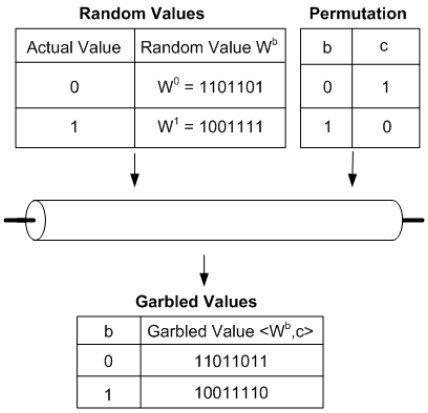

Figure 24.3: Garbled Circuit Wire

A gate table is then constructed for each Boolean gate in the circuit that can output the garbled value of the output wire when given the garbled input values. Finally, a table is constructed to map the garbled outputs of the circuit to the actual outputs. Construction of the gate tables and output tables requires knowledge of the random values and the random permutation assigned to the wires which are kept private by the garbled circuit constructor. Without knowledge of these random values and permutation, it is hard to work out any intermediate values in the circuit. To execute the circuit, the garbled input values for the circuit are needed. The garbled inputs are found using a protocol called verifiable proxy oblivious transfer (VPOT) [23]. Once the garbled input values are known, the gate tables are used to find the garbled outputs by applying the XOR function over the output of a random function on the garbled inputs and the entry in the gate table. When all the garbled outputs have been calculated, the output tables are used to work out the actual output.

Worked Example

Figure 24.1 illustrates a small garbled circuit with an AND and an OR gate. The garbled circuit creator produces 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 $<W^b,c>$ so for wire *Z* the garbled value of *0* is $<W^0,C_0>=<01,0>=010$. We set the input values to *V*=1, *W*=1, and *Y*=0. With these input values, the output of this circuit should be 1.

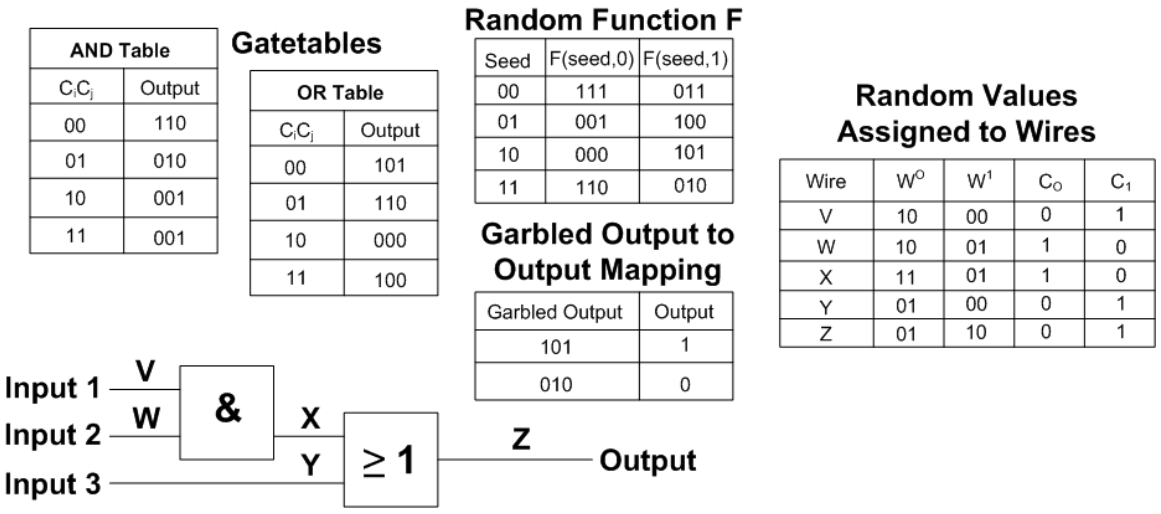

Figure 24.2 A simple garbled circuit

To execute this circuit the circuit executor would take the following steps:

1. Find out the garbled input values. 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.
2. Now we need to execute the gates. To execute the AND gate we use our garbled inputs and the gatetable. The equation used to calculate the garbled output is $\text{GarbledOutput}=F(W^b,c_j)\oplus F(W^b,c_i)\oplus \text{Gatetable}(c_i,c_j)$. So, the output is 001 ⊕ 111 ⊕100 = 010.

3. Now we need to execute the OR gate. The output is 101 ⊕ 001 ⊕ 001 = 101. Using the garbled output to output mapping we can see the output of the garbled circuit is *1*.

Note: a real circuit that executes an auction has many thousands of gates.

Garbled Circuits for Auctions

Naor, Pinkas, and Sumner [8] introduced the two party garbled circuit based protocol for a sealed-bid, two-server auction using oblivious transfer. The protocol features an auctioneer and an auction issuer as the two parties. This protocol is an efficient single pass protocol that also preserves the communication pattern of traditional auctions, see Figure 24.2.

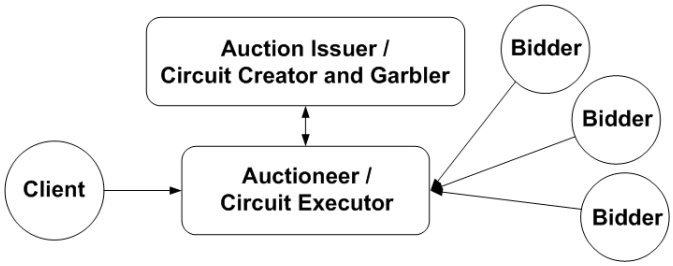

Figure 24.3 Garbled Circuit Interaction Diagram.

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. In addition, bidders do not have to encrypt values, which can be computationally expensive.

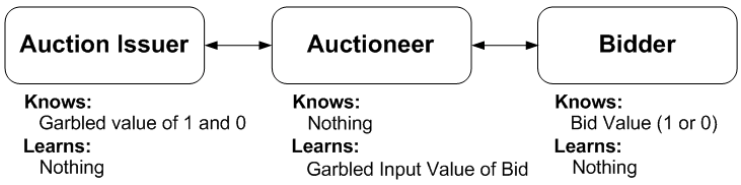

Figure 24.4 Views for Each Party.

Figure 24.4 shows the views for each party when conducting the protocol to learn the garbled input values for the auction circuit.

24.3.1.4 The Garbled Circuit Auction Protocol

The basic steps of a Sealed-Bid auction using the garbled auction 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 bits in the price, and the auction issuer being used.
- Bidders register to take part in the auction.
- Auctioneer requests a garbled circuit from the auction issuer giving the auction issuer the number of bidders, goods, and number of bits in the price.
- 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 auctioneer receives the garbled circuit and output mapping.
- The auction issuer, auctioneer, and bidders use the VPOT protocol 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.

24.3.2 Polynomial

The idea of using polynomials to efficiently share a secret between a group of people originated with Shamir [cite] in his seminal paper *How to Share a Secret*. The following question set by Liu [cite] captures the essence of the problem:

> *Eleven scientists are working on secret project. They wish to lock up the documents in a cabinet so that the cabinet can be opened if and only if six or more of the scientists are present. What is the smallest number of locks needed? What is the smallest number of keys each scientist must carry?*

The answer to this is 462 locks and 252 keys per scientist. Clearly this is not a practical situation and scales exponentially as the number of scientists increases. This is a (k, n) threshold scheme, where we divide the secured data D into n pieces and require at least k of those pieces to reconstruct the original. The choice of k and n determine the strength of the system, and of course $k < n$. Such threshold schemes are ideal when we wish to establish cooperation within a group of mutually un-trusted and possibly competing entities. Shamir's breakthrough was to solve this problem efficiently using polynomials. Shamir's algorithm is based on the idea that a polynomial of degree $k-1$ can be uniquely identified by k points. Therefore we can construct a polynomial $f(x)$ of degree $k-1$ using $k-1$ random coefficients and the secret D as the constant. We then pick n random x values and solve the polynomial creating n points (or key shares) of the form (x_i,y_i). To retrieve the value D we need at least k of these shares (any combination) and we can then reconstruct the coefficients using Lagrange interpolation.

Using polynomials for an auction protocol, particularly a combinatorial one, does require some modification of the original scheme. In this case the secret bid is stored in the degree of the polynomial rather than in the constant [yokoo-poly]. This has the nice property that two secret bids (encoded as polynomials A and B) can be compared using only local information with the following formulae: $max\{deg(A), deg(B)\} = deg(A + B)$,

$deg(A) + deg(B) = deg(A \cdot B)$. Hence, each auction evaluator (with key share *ks*) can compute locally its share of $A + B$ by computing $A(x_{ks})+B(x_{ks})$, or of $A \cdot B$ by taking the product $A(x_{ks}) \cdot B(x_{ks})$.

The Polynomial Auction Protocol

This protocol was developed by Yokoo and Suzuki [11] and is more fully described in [24] with implementation performance results. It is outlined here in the following five steps:

1. The client publishes information about goods under auction with an auction graph. Also published is a set of evaluator resolve values by *E* evaluators for generating polynomial shares, a constant value *c* for weight resolution, and a threshold value *t* is published that is used to prevent interference from less than *t* corrupt auctioneers.

2. Bidders calculate valuations and generate weights for each of the edges from the auction graph they are interested in. To form a weight *w* a threshold modifier $t \cdot (j - i)$ is added to the valuation, where *j* is the identification number of the larger node and *i* is the identification number of the smaller node. This modifier ensures that t evaluators are required to decrypt any edge and it is weighted against node identifiers so that costs are consistent over paths of different lengths. A random polynomial of degree *w* and constant *0* is solved with each evaluator's resolve value and sent to the respective evaluator to be recorded as an edge on their copy of the auction graph.

3. The optimal value of the auction is the sum of bidder valuations on the greatest path through the graph. As bids are distributed amongst the evaluators, valuations must be reconstructed by interpolating shares published by evaluators. The cost of node n_0, denoted $f(0)$ is the sum of all paths in the graph, which each evaluator calculates using dynamic programming and publishes.

4. Evaluators use binary search to collaboratively discover the optimal value o. Using Lagrange interpolation a polynomial can be recovered with a degree of one less than the number of shares used, and it is this number which the search discovers. At each iteration, evaluators publish shares masked with shares from M mask publishers generated in a similar ways as bids. The polynomial degree for the mask is equal to the number of shares to use and the constant equal to c. If the Lagrange polynomial has a constant equal to $M \cdot c$, the optimal value is less than or equal to one less than the number of shares, otherwise the optimal value is greater than the number of shares. The optimal value d is therefore found where $d \leq o$ and $d + 1 > o$.

5. An optimal path is traced from n_0 to n_{N-1}. As the cost of an optimal node n_i is the greatest path cost from n_i to n_0, the cost of an edge added to $f(j)$ of connected node j is $f(i)$. Beginning with $i = 0$ an optimal path is found by iteratively checking whether $f(j) + f(n_i, n_j) > f(i) - 1$. A comparison is carried out as in step four: $f(i) - 1$ evaluators publish masked shares of $f(j) + f(n_i, n_j)$ and interpolation is used to check for a constant not equal to Mc. When an optimal node n_j is found, binary search is used to determine $f(j)$ and the cost of the previous optimal edge is the difference between the two nodes $f(i) - f(j)$.

Worked example

Ten evaluators $\{e_1, e_2 \ldots e_9, e_{10}\}$ publish resolve values $\{1, 2, 3 \ldots 9, 10\}$. The client publishes details of two goods *G1* and G*2* with the initial graph as per section *24.3.1* and the constant value $c = 100$. The threshold is set as *1* for simplicity of examples, but note that this compromises security by removing threshold properties. The initial graph with corresponding valuations is shown in *figure 24.7*.

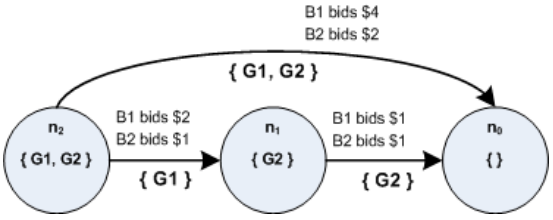

Figure 24.5 The initial auction graph with corresponding valuations.

Complete valuations and bid weights for two bidders B_1 and B_2 are given in table ***24.1***. Bidder B_1 generates a set of polynomials with a fixed coefficient of one corresponding to the weights: $x^6 + x^5 + x^4 + x^3 + x^2 + x$, $x^2 + x$ and $x^2 + x$ for edges (n_2, n_0), (n_2, n_1) and (n_1, n_0) respectively. The resolved polynomials sent to evaluators for both bidders are shown in ***Table 24.2.*** The weakness of using a fixed coefficient of one is apparent, as bids for the same value are identical. With random coefficients, interpolation is still possible and bids for the same value will differ.

Edge	Goods	B_1		B_2	
		Valuation	*w*	Valuation	*W*
(n_2, n_0)	$\{g_1, g_2\}$	4	6	2	4
(n_2, n_1)	$\{g_2\}$	2	3	1	2
(n_1, n_0)	$\{g_1\}$	1	2	1	2

Table 24.1 Bidder valuations and weights for each of the goods combinations.

	B_1			B_2		
Evaluator	(n_2, n_1)	(n_1, n_0)	(n_2, n_0)	(n_2, n_1)	(n_1, n_0)	(n_2, n_0)
E_1	$1^3 + 1^2 + 1 = 3$	2	6	2	2	4
E_2	$2^3 + 2^2 + 2 = 14$	6	126	6	6	30
E_3	$3^3 + 3^2 + 3 = 39$	12	1092	12	12	118
…	…	…	…	…	…	...
E_4	$9^3 + 9^2 + 9 = 819$	90	597870	90	90	7380
E_5	$10^3 + 10^2 + 10 = 1110$	110	1111110	110	110	11110

Table 24.2 Bids for bidders B1 and B2.

The evaluators compute f-values for each node using their bid shares and publish $f(0)$. Evaluator e_0 finds $f(1) = 3 + 5 = 5$ and $f(0) = (2 + 2)\ 5 + (4 + 6) = 30$. The shares for all evaluators are provided in table *24.5*.

Evaluator	$f(2)$	$f(1)$	$f(0)$
e_1	0	5	30
e_2	0	20	396
e_3	0	51	2436
…	…	…	…
e_4	0	909	768870
e_5	0	1220	1390620

Table 24.3 Evaluator shares of the f-values.

The minimum optimal value is set to zero, the maximum to nine and so initially $d = 4$. Resolved random polynomials of degree four and constant of one hundred are sent to the evaluators from the mask publishers. Because random coefficients are not used in this example both mask publishes generate the same masking polynomial $x^4 + x^3 + x^2 + x$. The total masks received by each evaluator added to evaluator shares for f (0) are:

e_1	208 + 30 = 238
e_2	260 + 396 = 656
e_3	440 + 2436 = 2876

…	…
e_4	14960 + 768870 = 783830
e_5	22420 + 222420 = 1413040

The Lagrange polynomial of these five points is $1988x^4 - 15211x^3 + 46185x^2 - 60334x + 28480$. As the polynomial's constant is not equal to 200 $o > 4$. *Table 24.6* shows a complete binary search for the optimal value, determined to be six, using evaluator shares.

d	Polynomial	Constant	
4	$195x^4 - 1293x^3 + 3784x^2 - 4808x + 2360$	2360	$o > 4$
7	$2x^7 + 3x^6 + 5x^5 + 10x^4 + 12x^3 + 8x^2 + 4x + 200$	200	$o \leq 7$
5	$26x^5 - 165x^4 + 747x^3 - 1616x^2 + 1768x - 520$	-520	$o > 5$
6	$3x^6 + 5x^5 + 10x^4 + 12x^3 + 8x^2 + 4x + 200$	200	$o \leq 6$

Table 24.4 Evaluator shares of the f-values.

Once the complete optimal path has been traced to n_{N-1} the highest bid valuations can be computed by subtracting the threshold modifier from each respective edge cost.

The edges from n_0 are checked and all but (n_2, n_0) are less than or equal to $f(0) - 1$. Once this edge is found the search ends as the destination is n_2. The actual cost is $6 - 1 \cdot (2 - 0) = 4$ and the optimal allocation is published as B_0 wins $\{g_1, g_2\}$ for $4.

24.3.3 Homomorphic

There is much in common between this protocol and the previous polynomial example, they both are threshold trust schemes and encode the auction in a graph. However they use different cryptographic techniques: Polynomial encryption and Homomorphic encryption respectively. Homomorphic cryptosystems maintain an algebraic operation between cipher text and plaintext where the operation on the cipher text applies to the

plaintext accordingly. Current cryptosystems only provide one operation, usually addition or multiplication and this chapter uses a system with multiplication such as El-Gamal.

In recent years electronic elections have received much focus from cryptographers. The problem of computing an election outcome while maintaining voter anonymity is well known. The following version of the popular *Mafia* party game captures the essence of the problem:

> *The population is divided into the Mafia and the townspeople. The Mafia's goal is to rob the townspeople, while the Town's goal is to imprison the Mafia. The members of the Mafia are aware of each other's identities, however the townspeople are not aware of anything other than their own role. The game is played in two phases. In the first phase the Mafia discusses and then votes by secret ballot to choose the townsperson to rob. The victim is then eliminated from the game. In the second phase the entire town (including the Mafia) discuss who to imprison and then vote by secret ballot. The prisoner is eliminated from the game. The secret ballot allows the Mafia to misdirect the townspeople, and indeed the Mafia can likewise become involved in their own sub-game.*

We can build a simple voting protocol to preserve anonymity using Homomorphic encryption for small scale voting. A vector of Homomorphic cipher text is offered to voters for each option to be voted on, the maximum length of each vector being the maximum number of voters. Each element in a vector corresponds to a potential voter and starts as an encrypted zero. One at a time each voter retrieves the current vectors and removes an encrypted one from the right hand side of the vector they wish to vote for,

adding an encrypted value not equal to zero to the left hand side (called left-shifting). A required property of the cryptosystem is randomizability, where two encryptions of the same plaintext should not produce the same cipher text. Voters can therefore multiply each element in the vectors by a cipher text of one, to hide the modifications. Once voting has closed, vectors can be decrypted and the outcome determined by counting the number of elements not equal to zero. This provides a solution which computes the outcome without revealing individual votes.

Of course a corrupt official with the decryption key could decrypt two consecutive sets of votes and determine the difference. To ensure vectors are not decrypted prematurely, threshold encryption can be used so votes can only be opened by consensus. One complication that is not dealt with here is techniques to stop voters from cheating by incrementing multiple vectors or a vector by more than one. Assuming everyone voted, the votes will not add up to the expected numbers but it is not possible to determine who rigged the vote.

This Homomorphic vector approach can be applied to combinatorial auctions in a similar manner as polynomial encryption was used in polynomial auctions. Bids are represented by threshold Homomorphic vectors, where the value of the vector is a series of encrypted common values followed by encrypted ones. Evaluators are divided into groups which each evaluate a node in the auction graph. Secure Dynamic programming is performed by recursively evaluating nodes: each group left-shifts outgoing edges by the highest

incoming path cost, which is found by multiplying alternative incoming edges together. Bid vector preliminaries are provided before the protocol is described.

Bid Vector Representation

As already stated, Homomorphic bid vectors are made from a series of encrypted constants followed by a series of encrypted ones. Each element in the vector is encrypted with the public key PK^+ of the relevant evaluator group. The value of the vector is equal to the number of elements not equal to one. For example a bid of three in a vector with a maximum bid of five and constant Z encrypted with the key for an evaluator group 0, is represented by: $PK^+_0 (Z), PK^+_0 (Z), PK^+_0 (Z), PK^+_0 (1), PK^+_0 (1)$.

Bid Vector Comparison

Vectors need to be compared for alternate bidders on edges and alternate paths to nodes. Two vectors can not be directly compared without decrypting, however the highest of the two can be found using multiplication if they have been encrypted with the same key. For instance, if two bidders bid four and two for the same edge then they would be multiplied as follows:

$$
\begin{array}{ll}
 & PK^+_0 (Z), PK^+_0 (Z), PK^+_0 (Z), PK^+_0 (Z), PK^+_0 (1) \\
* & PK^+_0 (Z), PK^+_0 (Z), PK^+_0 (1), PK^+_0 (1), PK^+_0 (1) \\
\hline
 & \boldsymbol{PK^+_0 (Z^2), PK^+_0 (Z^2), PK^+_0 (Z), PK^+_0 (Z)}, PK^+_0 (1)
\end{array}
$$

The result shows that the higher of the two vectors is four.

It is also possible to check whether a vector represents a number greater than or equal to a given value without decrypting the entire vector. This is performed by decrypting just the element located at the given value and checking for plain text not equal one.

Bid Vector Addition

A constant value *v* can be added to an encrypted vector by left shifting the vector by the value required. *v* elements are removed from the right hand side, and *v* common values encrypted with the same key as the vector are added to the left hand side. The vector is randomized by multiplying the unchanged elements by an encrypted one. When a modified vector is added to the public bulletin board, randomization prevents participants from counting the number of changed elements.

The Homomorphic Auction Protocol

This auction protocol is another scheme designed by Suzuki and Yokoo [4]. The protocol is further described in [25] and includes measured performance results. It is described here in the following four steps:

1. An auction graph is generated and published in the same manner as for polynomial auctions. Evaluators are divided into groups of *E* evaluators and each group is assigned one of *N* nodes from the graph. Depending on security requirements, a threshold *t*, evaluators are required to decrypt bid vectors. For each node *v*, a shared public key PK_v^+ is generated along with *E* private keys PK_v^{-1}, PK_v^{-2}, PK_v^{-3} ... PK_v^{-E} for each evaluator. Information published for bidders is: the graph with public keys mapped to edges, the maximum bid vector length *B* and the common value *Z*.

2. Bidders value each combination of goods, generating bids for each edge they are interested in, submitting them to a public bulletin board. Each bid contains a

Homomorphic bid vector encrypted with public key PK^{+}_{v} of the edge (the shared public key of the destination node) with a reference to the edge.

Once bidding has closed, evaluation using dynamic programming begins by recursively discovering the optimal values of each node. The optimal value of a node *i*, *f(i)* is the sum of the edges on the highest cost path from the end node n_{N-1} to n_i. The optimal value of the graph is the optimal value of the root node n_0 which provides the total of the highest cost path in the graph. An optimal edge value is the optimal value of it's source node added to the highest bid vector for that edge. Although individual edge values are not discovered at this point, the highest incoming optimal edge value needs to be, and it is the decryption of the multiplicative sum of all incoming bid vectors left-shifted by the optimal value of their source nodes.

Node evaluators decrypt the highest incoming optimal edge value (the multiplicative sum of incoming edges) using their private keys, and publish their shares. Once *t* shares have been published, the optimal value of the node can be decrypted and it is added to all outgoing edges by left-shifting and randomizing.

3. The total optimal value is used to trace back the optimal path through the optimal bid vectors. An edge is on the optimal path, if it's optimal bid vector is equal to $d = f(v)$ where *v* is the edge's destination node. It is possible to discover this without releasing bid values of edges not on the winning path by only checking

position *d* for a value not equal to one. The process starts from the root node n_0 and works backwards towards n_{N-1}. When an optimal edge is found, it's source node is published and the incoming edges of that node are checked. When n_{N-1} is reached, the optimal path is found. The decryption is performed in the same manner as in step 3, with evaluators publishing shares of the bid vector position.

4. Once the optimal path has been traced back bidders for each optimal edge can be found. Using the original bid vectors for each of the optimal edges, the same technique used in step four to identify optimal edges is used to identify optimal bid vectors and hence the winning bidders. The position to check is the value that the edge provides, i.e. *f (destination) – f (source)*.

Worked example

An auction graph is created for two goods; *{g1, g2}* and is published with $B = 5$ and $Z = 3$. Bidders B_1 and B_2 generate and post valuations and bid vectors for each of the edges, given in table 24.7. The graph with corresponding valuations is shown in figure 24.8.

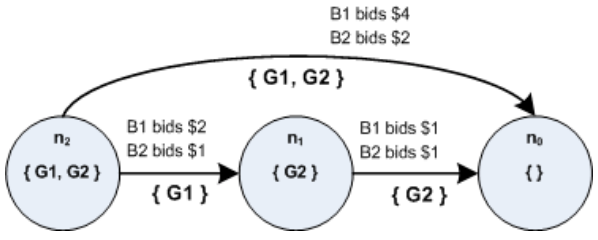

Figure 24.7 The initial auction graph with matching valuations.

	Edge	Goods	Valuation	Bid Vector

B_1	(n_2, n_0)	$\{g_1, g_2\}$	4	PK^+_0 (3), PK^+_0 (3), PK^+_0 (3), PK^+_0 (3), PK^+_0 (1)
	(n_2, n_1)	$\{g_2\}$	2	PK^+_0 (3), PK^+_0 (3), PK^+_0 (1), PK^+_0 (1), PK^+_0 (1)
	(n_1, n_0)	$\{g_1\}$	1	PK^+_0 (3), PK^+_0 (1), PK^+_0 (1), PK^+_0 (1), PK^+_0 (1)
B_2	(n_2, n_0)	$\{g_1, g_2\}$	2	PK^+_0 (3), PK^+_0 (3), PK^+_0 (1), PK^+_0 (1), PK^+_0 (1)
	(n_2, n_1)	$\{g_2\}$	1	PK^+_0 (3), PK^+_0 (1), PK^+_0 (1), PK^+_0 (1), PK^+_0 (1)
	(n_1, n_0)	$\{g_1\}$	1	PK^+_0 (3), PK^+_0 (1), PK^+_0 (1), PK^+_0 (1), PK^+_0 (1)

Table 24.5 Bids for B_1 and B_2.

As there are no incoming edges for n_2, $f(2) = 0$ and so the bid vectors on the outgoing edges from n_2 do not need to be left shifted by anything. The corresponding bid vectors are multiplied together:

$$\begin{array}{ll} & PK^+_0 (3), PK^+_0 (3), PK^+_0 (3), PK^+_0 (3), PK^+_0 (1) \\ * & \underline{PK^+_0 (3), PK^+_0 (3), PK^+_0 (1), PK^+_0 (1), PK^+_0 (1)} \\ & PK^+_0 (9), PK^+_0 (9), PK^+_0 (3), PK^+_0 (3), PK^+_0 (1) \end{array}$$

$$\begin{array}{ll} & PK^+_1 (3), PK^+_1 (3), PK^+_1 (1), PK^+_1 (1), PK^+_1 (1) \\ * & \underline{PK^+_1 (3), PK^+_1 (1), PK^+_1 (1), PK^+_1 (1), PK^+_1 (1)} \\ & PK^+_1 (9), PK^+_1 (3), PK^+_1 (1), PK^+_1 (1), PK^+_1 (1) \end{array}$$

These resulting vectors are published as the highest bid vectors from n_2 to n_0 and n_1 respectively. The evaluators for n_1 can now find the optimal value as the only incoming optimal bid vector has been published. The evaluators each use their private key to decrypt a share of the bid vector. Once t shares have been published, the optimal value is decrypted from the shares to find that $f(1) = 2$. The outgoing edge to n_0 is left shifted by 2, randomized and then published. The evaluators of n_0 multiply the two incoming edges together:

$$\begin{array}{ll} & PK^+_0 (3), PK^+_0 (3), PK^+_0 (3), PK^+_0 (3), PK^+_0 (1) \\ * & \underline{PK^+_0 (3), PK^+_0 (3), PK^+_0 (3), PK^+_0 (1), PK^+_0 (1)} \\ & PK^+_0 (9), PK^+_0 (9), PK^+_0 (9), PK^+_0 (3), PK^+_0 (1) \end{array}$$

And perform shared decryption, finding that $f(0) = 4$. This value is published as the optimal value of the graph.

Position four of n_0's incoming optimal bid vectors are checked for a value not equal to one. The evaluators each publish their shares and the decrypted values are found as *3* and *1* for edges (n_2, n_0) and (n_2, n_1) respectively. n_1 is published as not optimal, n_2 is published as optimal and because $n_2 = n_{N-1}$, the optimal path has been found and it is also published as the single edge (n_2, n_0).

Position *4 (4 - 0)* in edges (n_2, n_0) and (n_1, n_0) are checked for a value not equal than one. The value from b_1's vector is three and the value from b_2's vector is one so the result is published; b_1 having won both goods *{g1, g2}* for four dollars.

24.4 Comparison of Privacy Preserving Properties

Each of the three protocols introduced in this chapter differs in what privacy preserving properties are offered. For example privacy in Garbled circuits relies on only two parties (the Auctioneer and Issuer) not collaborating, while the other two schemes can require a variable number of parties to not collaborate. This section compares the properties that the schemes offer against Bob and Jane's problems outlined in Step 1 of section *24.1.1*. These problems are:

- Insider trading – the auctioneer misuses information about current auction state (i.e. current valuations) for competitive advantage. This information could be provided (possibly sold) to a bidder in a sealed bid auction so the bidder can bid the minimum valuation possible to win.

- Private information revelation – the auctioneer gathers a history of information such as minimum bids for use in future auctions. This information may be used for example to set unfair reserve prices, manipulating the pricing in a second priced auction.
- Bid filtering – the auctioneer drops bids based on personal interests.
- Lying auctioneer – The auctioneer lies about the outcome of the auction, either misreporting winners or prices.

This first step to securing auctions aims to solve the first three issues by sealing bids, however as bids are no longer publicly known, the final problem is introduced. All three protocols solve the insider trading problem as no bids are decrypted until the auction is over. Garbled Circuits prevents information leakage provided the two participants do not collude. This is effectively equivalent to a *(2, 2)* threshold scheme. The Polynomial and Homomorphic schemes both provide a variable *(t, n)* threshold scheme, but differ in what can be released. In Polynomial, the cost of each node can be determined without any decryption of edges, and so the longest path is found without leakage. However in the homomorphic protocol, because addition is not offered by the cryptosystem, evaluator groups must decrypt the multiplicative sum of incoming edges. It only requires one evaluator in each group to publish the maximum cost for that node. If two connected nodes have their maximum cost released then the maximum edge cost between them can be calculated. For an edge not on the optimal path, this is private information that should not be revealed.

There are two classifications of bid filtering: bids may be dropped dependant on their value or they may be dropped dependant on who made them. Auctioneers in each of the three protocols obviously cannot drop bids based on value before evaluation, and once evaluation starts, evaluators cannot drop bids without other evaluators noticing. In the garbled circuits protocol, the auction issuer can publish hash values of the data it receives from bidders that bidders can use to check that their bid was included\cite{garbled}. The polynomial and homomorphic protocols do nothing to stop bids being dropped based on who placed them, but they could be used with an anonymity service.

In the Garbled Circuit auction protocol, a corrupt auction issuer could create an invalid circuit that does not correctly compute the result of the auction. A corrupt auctioneer could also not even execute the circuit given to it by the auction issuer. There have been several methods suggested to verify these steps of the garbled circuit auction protocol [8]. In the Polynomial and Homomorphic protocols, it is not possible for an auctioneer to blatantly lie about the outcome at any of the decryption steps because evaluators can rerun final decryption steps. They can however publish fake shares which may affect the outcome if used. These may be able to be detected if enough combinations of different shares are used to check the outcome at each decryption. However, a better protocol would offer verification to ensure that active interference is discouraged and detected.

24.5 Comparison of Performance

Figure 24.7 shows the timeline for conducting an auction. The auction starts when the client requests the auctioneer to perform the auction, and ends when the winner of the

auction has been determined. There are 4 phases, auction creation, bidder registration, bid submission, and winner determination.

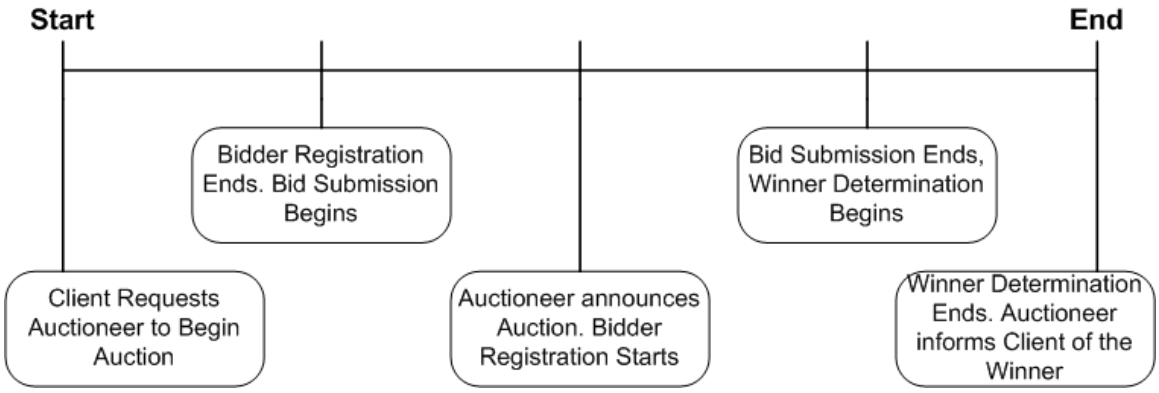

Figure 24.7 Auction Timeline

Each of the auction protocols uses different cryptographic techniques and these choices translate into different protocol performance characteristics. In the garbled circuit protocol there is potential for considerable communication overhead in transmitting the garbled circuit. In the polynomial protocol, there is potential for a high overhead when increasing the threshold of auctioneers needed to decrypt a value. In the homomorphic protocol potential overhead is increased by increasing the key size used in the encryption. This section quantifies the costs of these protocol decisions.

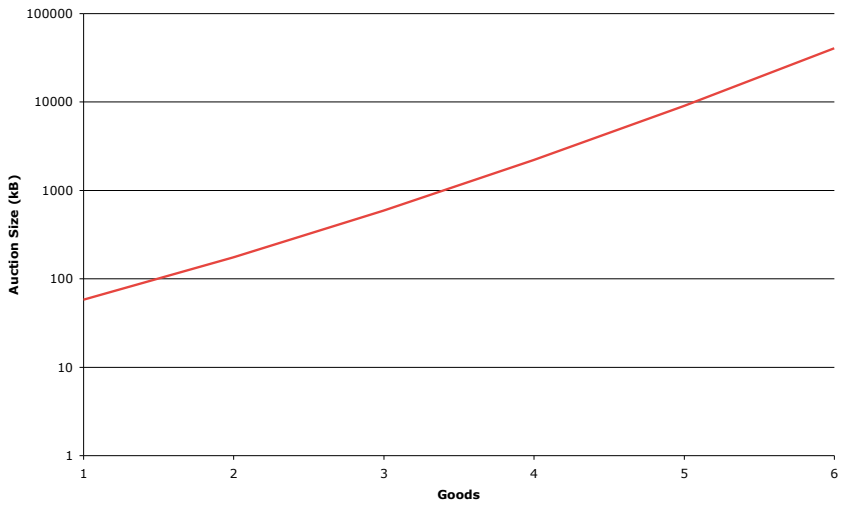

Figure 24.8 Size of Circuit vs Number of Goods

As the number of goods in an auction increases, so must the complexity of the garbled circuit. Figure 24.8 quantifies the relationship between the number of goods in the auction and the size of the garbled circuit. The size of the circuit increases exponentially with the number of goods. For an example auction with 3 goods, a maximum bid of 16, and 10 bidders the size of the circuit is about 600kB.

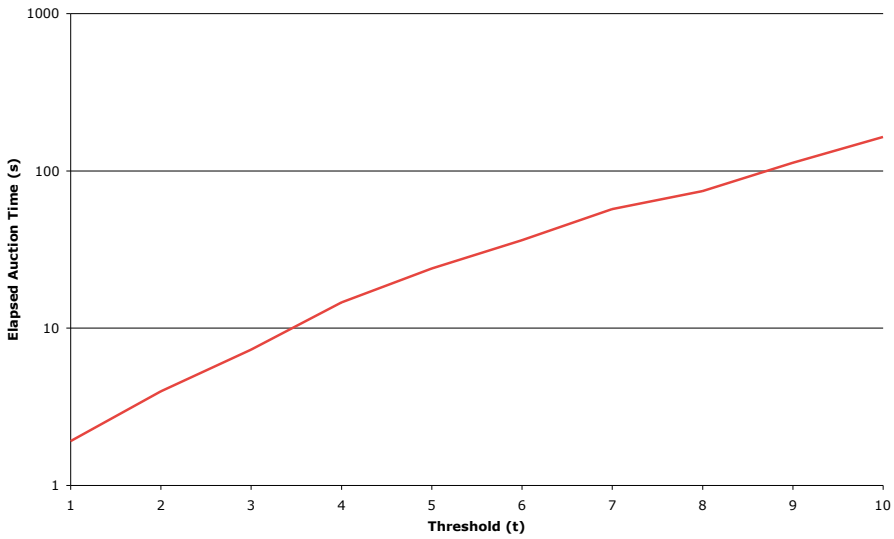

Figure 24.9 Polynomial Threshold Value vs Time

In order for the Polynomial auction to tolerate more corrupt auctioneers, the threshold must be increased (t =1, means we can detect 1 corrupt auctioneer). Figure 24.9 shows the exponetial effect of increasing this threshold. This exponential increase is due the time required to do the LaGrange interpolation as the degree of the polynomial storing the secret increases.

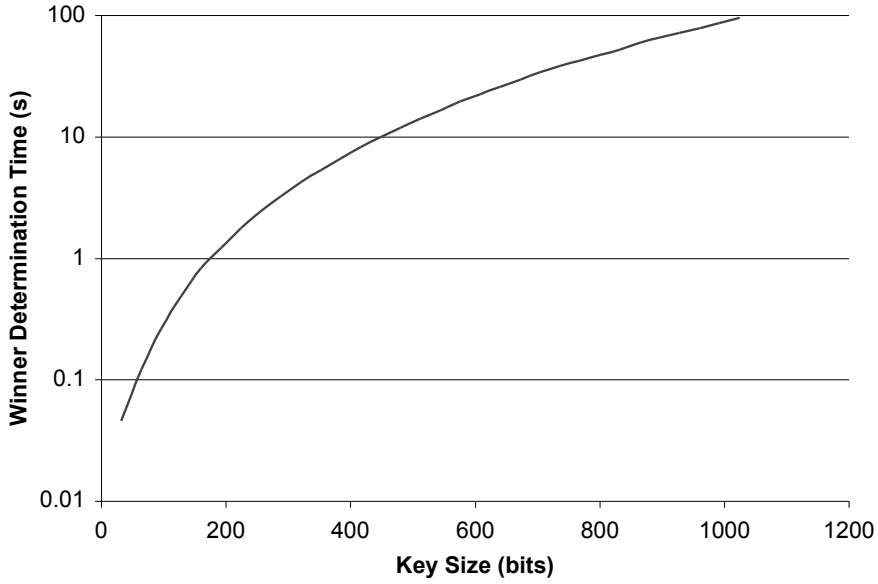

Figure 24.9 Homomorphic Key Size vs Time

To improve the security of the homomorphic scheme, the size of the encryption keys must increase. Figure 24.9 shows the effect of increasing the key size on the winner determination time. This is due in large to the additional overhead of factoring the larger primes needed for longer keys.

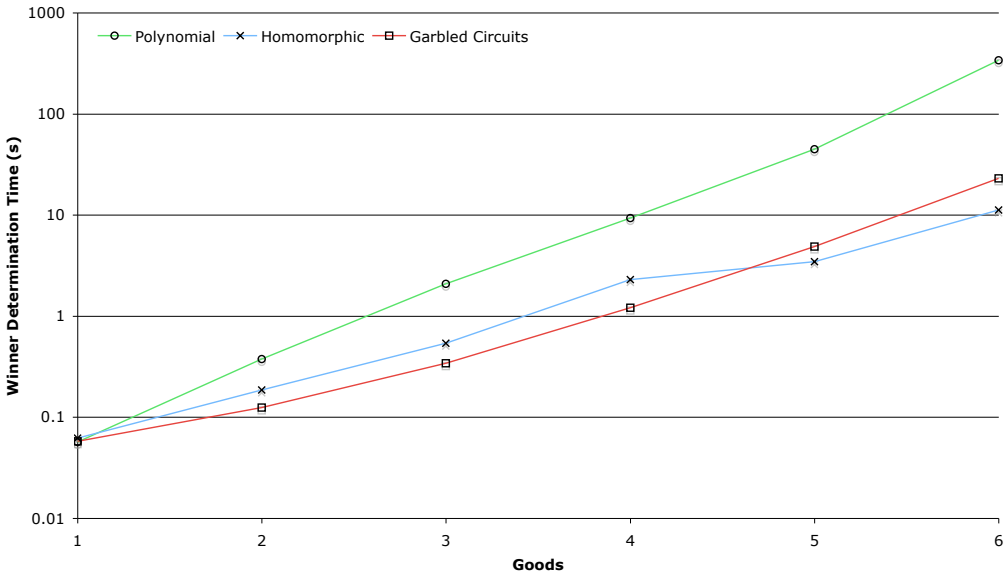

Figure 24.10 A performance comparison of the three auction protocols.

Finally, Figure 24.10 compares the performance of the three auction protocols for different numbers of goods. The garbled circuits protocol has the shortest winner determination time for number of goods less than 5, otherwise the homomorphic auction protocol does. The polynomial auction protocol has the worst performance of the three protocols examined in this chapter.

24.6 Verifiable Auction Protocols

The second step in securing auction protocols is to verify the outcome while maintaining privacy. Verification aims to detect cheating agents, determining that all bids are considered and the outcome is correct. This section introduces modifications to the homomorphic protocol, providing verification. Verification for use with privacy preserving auction protocols has been implemented using the following techniques:

- **Zero Knowledge Proofs:** used to prove some statement, without revealing any other information other than what is known before the proof was executed, see Goldwasser et. al [26].
- **Range Proofs:** used to prove that an encrypted value is the largest in a set of encrypted values and that an encrypted value is in a certain range [5].
- **Cut and Choose:** used in the garbled circuits protocol [8] where *x* copies of a garbled circuit are constructed for the auction and *x-1* randomly chosen copies are opened before the auction to check they have been correctly constructed.
- **Verifiable Secret Sharing:** used to verify shares of a secret have been correctly calculated [11].

The following verification protocol makes use of zero knowledge proofs to allow public verification of the auction process without revealing additional information other than what is known from the correct execution of the auction protocol.

Zero Knowledge Proofs

Zero knowledge proofs are used to prove some statement, without revealing any other information other than what is known before the proof was executed. A zero knowledge proof takes part between a prover and a verifier and typically consists of a commitment from the prover, a challenge from the verifier, and a response from the prover.

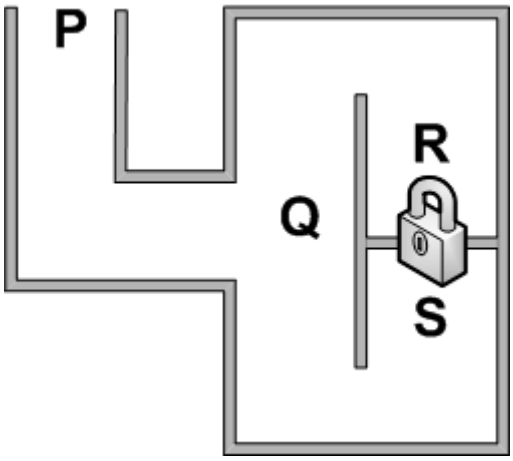

Figure 24.8: Ali Baba's Cave.

Figure 24.7 illustrates a famous zero knowledge proof known as Ali Baba's cave. Alice wants to convince the verifier Bob that she knows the secret password to open a door between *R* and *S*. To prove this while not revealing the secret password used, Alice and Bob conduct the following steps:

- Bob waits at P while Alice goes to either R or S (commitment).
- 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).
- Bob chooses either the top (R) or bottom (S) tunnel.
- Bob challenges Alice to come out of the tunnel of his choice. Alice can only exit the correct tunnel 100% of the time if she knows the password.
- If Alice does not know the secret words, there is a 50% chance she will come out from the wrong tunnel.
- Bob can then repeat this process as many times as he wants to convince himself that Alice knows the secret word, but Bob will never learn the secret word himself.

Outline

This verification process is divided into several operations:

- Prove a bid vector is valid. When a bidder submits a bid vector, the auctioneers and bidder prove that the bid vector is a valid bid vector.
- Prove a maximum bid from a set of bids. While calculating the result of the auction, auctioneers can prove that a bid is the maximum bid in a set of bids without revealing the value of any of the encrypted bids.
- Prove shift and randomize has been done correctly. The homomorphic protocol uses a technique called shift and randomize to add a constant to a bid vector, a proof can be published showing that this operation has been done correctly.

Proving a Bid Vector is Valid

Constructing an 'integrated' bid vector makes the proof easier and more efficient. This 'integrated' bid vector can then be converted to a standard bid vector as used in the homomorphic protocol. Suppose the bid vector is of length n and the bidder wants to bid value k. Then the i^{th} item of the integrated bid vector is of the form:

$$IBV_i = \begin{Bmatrix} PK^+(Z) \text{ if } i = k \\ PK^+(1) \text{ otherwise} \end{Bmatrix}$$

So for a bid vector of length $n=6$ and value $k=2$, the 'integrated' bid vector is:

$IBV = PK^+(1), PK^+(Z), PK^+(1), PK^+(1), PK^+(1), PK^+(1).$

Using this 'integrated' bid vector we can prove the bid vector is valid by using zero knowledge proofs that every item in the bid vector is an encryption of *1* or a *Z* [27] and that the product of every item in the bid vector decrypts to *Z* [28].

The encrypted 'integrated' bid vector is then converted to a standard bid vector by performing the following operations on the items in the bid vector.

$$BV_i = \begin{cases} IBV_i \text{ if } i = n \\ BV_{i+1} \times IBV_i \text{ otherwise} \end{cases}$$

So $IBV = PK^+(1), PK^+(Z), PK^+(1), PK^+(1), PK^+(1), PK^+(1)$ becomes

$BV = PK^+(Z), PK^+(Z), PK^+(1), PK^+(1), PK^+(1), PK^+(1)$.

To verify the bid vector is valid, a verifier will check the proofs that every item in the 'integrated' bid vector is *1* or *Z*, as well as the proof that the decryption of the product of all the items is *Z*. The verifier can then check that the standard bid vector was constructed correctly due to the homomorphic nature of the encryption used.

Proving a Maximum Bid from a Set of Bids

To prove one encrypted bid vector is the maximum from a set of encrypted bid vectors, an auctioneer publishes a verifiable shuffle of the set of encrypted bid vectors using techniques from [29]. Bid vectors should all be shuffled using the same permutation that is known only to the auctioneer. The auctioneer then proves that one of the entries in the maximum bid vector decrypts to *Z* while all other bids decrypt to *1* at the same index in the shuffled bid vectors. The bid vectors can be shuffled in such a way that the first item in the shuffled bids decrypts to *Z* for only one of the bids. The auctioneer can arrange the shuffling this way as they know the maximum bid value before they do the shuffle.

As all the other bids are shown to decrypt to *1* and we can verify the shuffle of the encrypted values this proves the maximum bid from the set of encrypted bid vectors without revealing the value of the maximum bid.

For example, suppose we have three bids:

Bid 1: $PK^+(Z), PK^+(Z), PK^+(Z), PK^+(1)$.

Bid 2: $PK^+(Z), PK^+(Z), PK^+(1), PK^+(1)$.

Bid 3: $PK^+(1), PK^+(1), PK^+(1), PK^+(1)$.

Bid 1 is the maximum bid. To prove this an auctioneer applies a permutation π to the bids say $\pi = \{3, 4, 1, 2\}$. The modified vectors are:

Shuffled Bid 1: $PK^+(Z), PK^+(1), PK^+(Z), PK^+(Z)$.

Shuffled Bid 2: $PK^+(1), PK^+(1), PK^+(Z), PK^+(Z)$.

Shuffled Bid 3: $PK^+(1), PK^+(1), PK^+(1), PK^+(1)$.

The auctioneer/s then prove that this shuffle has been done correctly [29] and that the elements at position *0* decrypt to *Z*, *1*, and *1* respectively.

Putting It All Together

This section outlines the process for a verifiable homomorphic combinatorial auction. Figure 24.8 illustrates a simple auction with two bidders bidding for two goods. The winner of the auction should be Bidder 1 who has the maximum bid of $3 for the two items together.

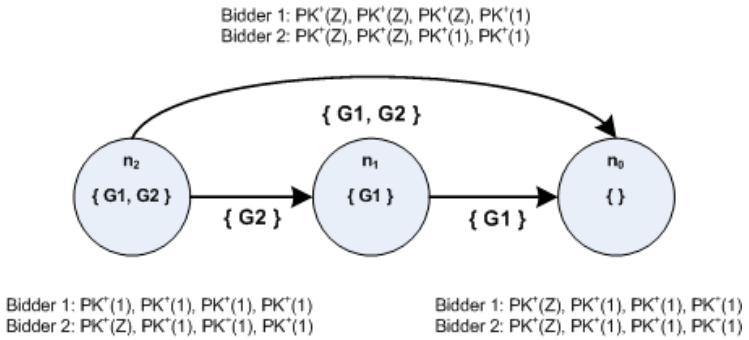

Figure 24.9 Example Auction.

The execution of the protocol involves the following steps:

- Bidders publish their bids for every. Bidders also publish proof that every item in their bid vector decrypts to a *1* or a *Z*.
- Auctioneers publish a proof that the product of all the items in the 'differentiated' bid vector decrypts to *Z*.
- Auctioneers conduct the auction. If they need to add constants to bid vectors they publish a proof this is done correctly using the proof in Section 'Proving shift and randomise' above.
- The auctioneers have found and published the optimal path and the winning bids in the optimal path. They now prove that the optimal path is indeed optimal and that the winning bids were the maximum using the proof from the section 'Proving a maximum bid from a set of bids' above.

24.7 Summary

When Grids cross organizational boundaries to become ad-hoc collections of resources with multiple owners, we strike the problem of establishing how we can trust other

organizations and their software during resource allocation. This is especially true in on-demand Grids and Market Oriented Grid or Utility computing, where such allocation decisions will have very real financial implications. Privacy preserving auction protocols prevent an auctioneer from identifying bidders and acting to advantage, or to disadvantage certain bidders. These auctions also prevent the auctioneer from stealing and distributing commercially sensitive bid information. Verifiable auction protocols allow independent verification of an auctioneer's actions and therefore provide cryptographic guarantees that the outcome of the auction includes all bids and that the outcome of the auction was computed correctly. In effect the auctions become on-demand auditable by any participant rather than operating as black boxes. Combining both privacy preserving and verifiable auction protocols removes the need to place trust in any auctioneer – be they actually trustworthy or not.

****

1. S. Fay, *The Collapse of Barings*, Richard Cohen Books, London, 1996.
2. H. R. Varian, Economic Mechanism Design for Computerized Agents, in *Proceedings of Usenix Workshop on Electronic Commerce*, 1995.
3. W. Vickrey, Counterspeculation, Auctions, and Competitive Sealed Tenders, *The Journal of Finance*, 16(1): pp. 8-37, 1961.
4. K. Suzuki and M. Yokoo, Secure Generalized Vickery Auction Using Homomorphic Encryption, in *Proceedings of the 7th International Conference on Financial Cryptography*, Guadeloupe, French West Indies, 2003.
5. H. Lipmaa, N. Asokan, and V. Niemi, Secure Vickrey Auctions without Threshold Trust, in *Proceedings of the 6th International Conference on Financial Cryptography*, 2002.
6. M. Harkavy, J. D. Tygar, and H. Kikuchi, Electronic Auctions with Private Bids, in *Proceedings of the Third USENIX Workshop on Electronic Commerce*, Boston, Massachusetts, USA, 1998.

7. C. Cachin, Efficient Private Bidding and Auctions with an Oblivious Third Party, in *Proceedings of the 6th ACM conference on Computer and communications security*, Kent Ridge Digital Labs, Singapore, 1999.
8. M. Naor, B. Pinkas, and R. Sumner, Privacy Preserving Auctions and Mechanism Design, in *Proceedings of the 1st ACM conference on Electronic Commerce*, Denver, Colorado, United States, 1999.
9. O. Baudro and J. Stern, Non-Interactive Private Auctions, in *Proceedings of the 5th International Financial Cryptography Conference*, Grand Cayman, BWI., 2001.
10. H. Kikuchi, (M + 1)St-Price Auction Protocol, *in Proceedings of the 5th International Conference on Financial Cryptography*, Grand Cayman, 2001.
11. K. Suzuki and M. Yokoo, Secure Combinatorial Auctions by Dynamic Programming with Polynomial Secret Sharing, in *Proceedings of the 6th International Conference on Financial Cryptography*, Southampton, Bermuda, 2002.
12. 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, 2002.
13. K. Peng, C. Boyd, E. Dawson, and K. Viswanathan, Robust, Privacy Protecting and Publicly Verifiable Sealed-Bid Auction, in *Proceedings of the 2nd International Conference on Information, Communications and Signal Processing*, 2002.
14. K. Peng, C. Boyd, E. Dawson, and K. Viswanathan, Five Sealed-Bid Auction Models, in *Proceedings of the Australasian information security workshop conference on ACSW frontiers*, Adelaide, Australia, 2003.
15. W. Ham, K. Kim, and H. Imai, Yet Another Strong Sealed-Bid Auctions, in *Proceedings of the Symposium on Cryptography and Information Security (SCIS)*, 2003.
16. X. Chen, B. Lee, and K. Kim, Receipt-Free Electronic Auction Schemes Using Homomorphic Encryption, in *Proceedings of the 6th International Conference on Information Security and Cryptology* 2003.
17. J. Nzouonta, An Algorithm for Clearing Combinatorial Markets in *Technical Report No. CS-2003-23*, Florida Institute of Technology, 2003.
18. K. Peng, C. Boyd, and E. Dawson, A Multiplicative Homomorphic Sealed-Bid Auction Based on Goldwasser-Micali Encryption, in *Proceedings of the 8th International Conference on Information Security*, 2005.
19. D. C. Parkes, M. O. Rabin, S. M. Shieber, and C. A. Thorpe, Practical Secrecy-Preserving, Verifiably Correct and Trustworthy Auctions, in *Proceedings of the 8th international conference on Electronic commerce*, 2006.
20. F. Brandt, How to Obtain Full Privacy in Auctions*, International Journal of Information Security*, 5(4): pp. 201-261, 2006.
21. B. Palmer, I. Welch, and K. Bubendorfer, Adding Verification to a Privacy Preserving Combinatorial Auction, in *CS-TR-07-3*, Victoria University of Wellington, 2007.

22. A. C. Yao, How to Generate and Exchange Secrets, in *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, 1986.
23. A. Juels and M. Szydlo, *A Two-Server, Sealed-Bid Auction Protocol*, in *6th Financial Cryptography Conference*. 2002.
24. K. Bubendorfer and W. Thomson, Resource Managment Using Untrusted Auctioneers in a Grid Econonmy, in *Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing*, Amsterdam, Holland, 2006.
25. K. Bubendorfer, I. Welch, and B. Chard, Trustworthy Auctions for Grid-Style Economies, in *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid (CCGrid06)*, 2006.
26. S. Goldwasser, S. Micali, and C. Rackoff, The Knowledge Complexity of Interactive Proof Systems*, SIAM Journal of Computing*, 18(1): pp. 186--208, 1989.
27. R. Cramer, R. Gennaro, and B. Schoenmakers, A Secure and Optimally Efficient Multi-Authority Election Scheme, in *Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology (EUROCRYPT)*, 1997.
28. D. Chaum and T. P. Pedersen, Wallet Databases with Observers, in *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, 1993.
29. J. Furukawa and K. Sako, An Efficient Scheme for Proving a Shuffle, in *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, 2001.