VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering
and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

Verifying Privacy Preserving Combinatorial Auctions

by

Ben Palmer

A thesis

submitted to the Victoria University of Wellington
in fulfilment of the requirements
for the degree of
Master of Science
in Computer Science.

Victoria University of Wellington
2009

Abstract

Suppose you are competing in an online sealed bid auction for some goods. How do you know the auction result can be trusted? The auction site could be performing actions that support its own commercial interests by blocking certain bidders or even reporting incorrect winning prices. This problem is magnified when the auctioneer is an unknown party and the auctions are for high value items. The incentive for the auctioneer to cheat can be high as they could stand to make a significant profit by inflating winning prices or by being paid by a certain bidder to announce them the winner. Verification of auction results provides confidence in the auction result by making it computationally infeasible for an auction participant to cheat and not get caught. This thesis examines the construction of verifiable privacy preserving combinatorial auction protocols. Two verifiable privacy preserving combinatorial auction protocols are produced by extending existing auction protocols.

Acknowledgments

I would like to thank first my supervisors Kris Bubendorfer and Ian Welch for their help thoughout this thesis. I would also like to thank Wayne Thomson for his help getting my tests running in the GAF and general help in understanding the GAF. Finally, I would like to thank my family and my partner Bree Kurtovich for constant support and ideas.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Online auctions have grown into a widely accepted way of trading goods and services. Auctions are a high profile method for consumer goods to be traded with the New Zealand auction site TradeMe having over one million items for sale, over one and a half million members, and has almost four hundred thousand visitors every day [50]. Worldwide auction site eBay has 233 million registered users that trade $1,839 US dollars worth of goods every second [14]. However, its not just consumer goods that are being traded using online auctions. In the USA, the Federal Communications Commission uses online auctions to sell licenses for electromagnetic spectrum [15] and, in the UK, the Office of Communications intends to use an online auction to sell licenses for 215MHz of spectrum [40]. Online auctions are a pervasive mechanism used by individuals, businesses, and governments to trade goods and services.

Combinatorial auctions are a special type of auction where bidders can place bids on combinations of goods as opposed to a single good auction. This enables bidders to take advantage of any synergistic value of goods. In a three good auction, a bidder could place a bid on good 1 only if they can also get good 2, so good 1 and good 2 together. Online combinatorial auctions have been used extensively to allocate truckload transportation in the United States where, by 2003, half a dozen software packages were available to facilitate the combinatorial auction of trucking routes [12]. These include OptiBid [35], Transportation Bid Collaborator [27], and others. The average annual value of transportation services auctioned in the period from 1997 to 2001 was $175 million US Dollars [12]. Online combinatorial auctions also have been used extensively for industrial procurement auctions. Mars Incorporated found a forty minute auction replaced a

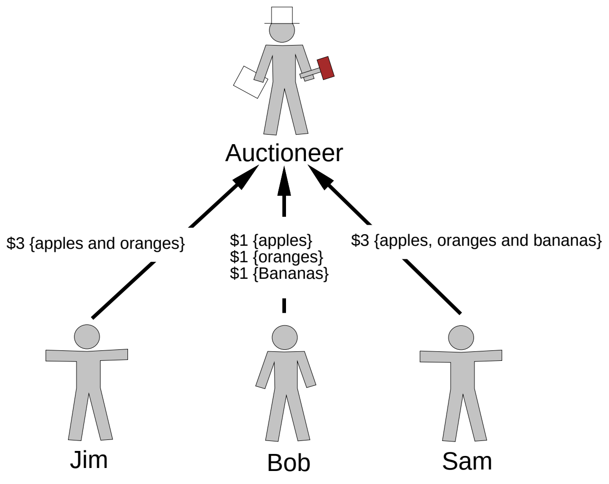


Figure 1.1: Example Combinatorial Auction

negotiation process that lasted over two weeks and required nine separate plane trips [12].

Figure 1.1 shows an example of a combinatorial auction where Jim wants apples only if he can also have oranges, Bob wants any subset of the goods, while Sam needs all three of the goods. The highest revenue for the auctioneer ($4) is generated by allocating the apples and oranges to Jim and the bananas to Bob.

1.1 The Auction Trust Problem

Suppose Alice is running a sealed bid auction of artwork for a charity organisation. She plans to hold the auction on her web site hosted by Sam. Bob and Jim submit bids to the auctioneer as shown in Figure 1.2.

There are several potential problems with this auction:

- Either Sam or Alice can peek at the bids. Many bidders prefer their bids to remain private especially in a competitive environment where bids are commercially sensitive information, alternately the bid values could be passed on to a competing bidder so they can make sure their bid is slightly higher.

- Alice could refuse to count certain bidders in the auction. Alice could act in collusion with a malicious bidder to make sure a competing bidder never has a bid counted in the auction.

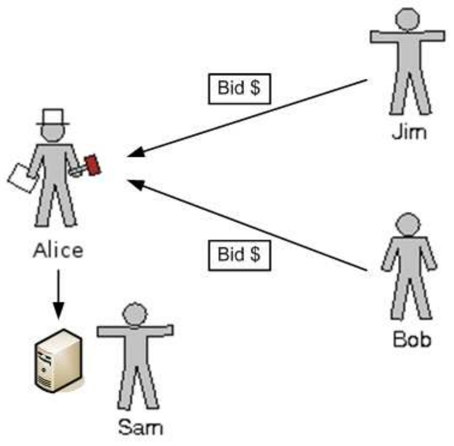


Figure 1.2: Art Auction

- Alice could arbitrarily choose a winner regardless of the bid values, the motivation for this is especially high if Alice is colluding with the bidder chosen as the winner.

- Alice could easily defraud the charity organisation by reporting a reduced winning price, and taking the difference herself.

A large amount of trust is placed in Alice with no way of checking whether she has correctly executed the auction. In current systems this trust is often placed in a central organisation such as TradeMe or the Federal Communications Commission.

Alice can be prevented from breaking privacy guarantees by using a privacy preserving auction where the values of bids are hidden using encryption or obfuscation yet can still be compared to find the winner. Figure 1.3 shows Alice holding a privacy preserving sealed bid auction on her web site hosted by Sam. Bob and Jim submit encrypted bids to the auctioneer.

In this auction, Alice and Sam are prevented from being able to peek at bids due to the obfuscation of bid values. This prevents sensitive information leaking to competing bidders. There has been a large amount of work in the area of secure auctions, starting with the work by Franklin and Reiter [16]. Auction protocols have been developed that are capable of conducting auctions involving multiple goods and bidders all while using cryptographic techniques to keep losing bid values secret from both auctioneers and other bidders.

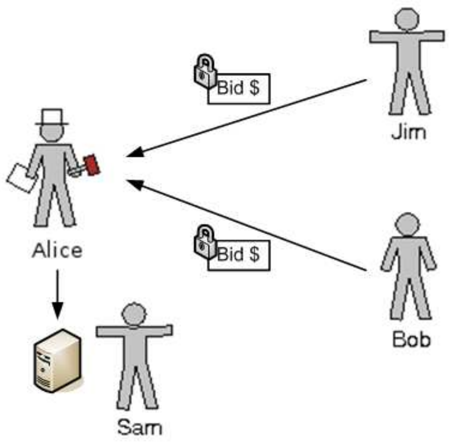


Figure 1.3: Privacy Preserving Art Auction

The possible auction attacks informally discussed so far can be divided in to the following categories:

- 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 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 to set unfair reserve prices or sold to some interested party.

- Bid filtering: The auctioneer drops bids based on personal interests.

- Malicious auctioneer: The auctioneer calculates an incorrect auction result. The auctioneer could announce a bidder as the winner regardless of bid values, or force the auction to calculate incorrect winning price/s or bidder/s.

Privacy preserving auction protocols address insider trading and private information revelation. By using encrypted bids that cannot be opened until after the end of the bid submission phase, the auctioneer is prevented from insider trading. The auctioneer either cannot open the encrypted bids, or when it can open the bids, it is too late to use the information on the auction state. Privacy preserving auctions keep losing bid values secret which prevents information revelation other than the information that has to be made public, such as the winning bidders and prices.

Verification prevents bid filtering and a malicious auctioneer. In a verifiable auction, bidders check that their bids have been counted in the auction to prevent bid filtering. Bidders can also check that the auction process has executed correctly to prevent a malicious auctioneer.

1.2 Security Goals

Verification protocols for privacy preserving auctions have the following security goals:

1. Bidders, auctioneers, or any third party should be able to verify the actions of the participants in the auction protocol giving a high confidence that the auction participants have correctly executed the auction.

2. Verification of the auction protocol should reveal no information other than what is revealed by the auction protocol.

3. It should be computationally infeasible for a bidder to submit an invalid bid that passes the verification checks.

4. It should be computationally infeasible for an auctioneer to not count all the bids and pass the verification checks.

5. It should be computationally infeasible for an auctioneer to announce an incorrect winning bidder(s) or price(s) and pass the verification checks.

1.3 Thesis Goals

The goal of this thesis is to create a verifiable privacy preserving combinatorial auction protocol. Privacy preserving auction protocols prevent insider trading and private information revelation and verifiable auction protocols prevent bid filtering and a malicious auctioneer. Combinatorial auctions are a useful tool for trading goods and services as shown by the examples presented in the introduction. There has been a significant amount of research work done on secure auctions starting with the work by Franklin and Reiter [16] and this thesis aims to investigate this work, identify solutions or techniques, and use them to build a verifiable privacy preserving combinatorial auction protocol.

1.4 Contributions

The main contributions of this thesis are:

1. Producing a taxonomy of current privacy preserving auction protocols providing a useful way to classify and discuss current solutions. The taxonomy of cryptographically secure auctions produced for this thesis reveals that there is no existing privacy preserving auction protocol for combinatorial auctions. Two possible alternatives suggest themselves:
    (a) Extend an existing verifiable privacy preserving auction protocol to compute combinatorial auctions.
    (b) Extend an existing privacy preserving combinatorial auction protocol to be verifiable.
2. For alternative 1a the existing verifiable privacy preserving auction protocol by Naor, Pinkas, and Sumner called garbled circuits [36] was identified and extended to compute combinatorial auctions. This involved:
    (a) The creation of a novel Boolean circuit that enables combinatorial auctions to be conducted by the garbled circuits auction protocol. Previously garbled circuits had only been used to conduct first price and (M+1) priced auctions,
    (b) The implementation of the garbled circuits protocol and the combinatorial Boolean circuit, and
    (c) A performance analysis showing the performance of the garbled circuit auction protocol when conducting combinatorial auctions.
3. For alternative 1b an existing privacy preserving combinatorial auction protocol by Suzuki and Yokoo [53] was identified and extended to be verifiable using zero knowledge proofs. This involved:
    (a) The design of a public and group verification protocol using existing zero knowledge proofs to verify the various steps of the auction.

The public verification protocol makes use of existing zero knowledge proofs and combines them in a novel way,

(b) The implementation of the group verification protocol and applying it to an existing implementation of the privacy preserving auction protocol,

(c) A security analysis and tests to confirm that the group verification protocol does indeed detect a party not keeping to the correct auction protocol, and

(d) A performance analysis showing the overhead of the group verification protocol when compared to the performance of the original protocol.

1.5 Thesis Organisation

This thesis begins by presenting a taxonomy of existing secure auction protocols in Chapter 2 to detail the related work in this area. Chapter 3 goes into more detail about zero knowledge proofs as well as presenting a detailed look at the auction protocol by Suzuki and Yokoo [53]. The work extending garbled circuits to compute combinatorial auctions is in Chapter 4. The group verification protocol to verify the result of the auction protocol by Suzuki and Yokoo is presented in Chapter 5. Chapter 6 details the public verification protocol to verify the result of the auction protocol by Suzuki and Yokoo. An analysis of the security of these verification protocols is presented in Chapter 7 and an analysis of the performance of the auction protocols is presented in Chapter 7. Conclusions and future work complete this thesis in Chapter 8.

Chapter 2

Secure Auction Taxonomy

Peng, Boyd, Dawson and Viswanathan have published their view of desirable properties of privacy preserving online auctions [44] that can be used to classify and discuss different auction protocols:

- Correctness. The auction winner(s) and the winning price(s) is calculated correctly.
- Confidentiality. Bids are kept private before the bid opening phase.
- Fairness. A submitted bid cannot be modified, or denied.
- Price Flexibility. Allow bidders to bid any amount between the minimum and maximum price and not just an item in a limited set of allowed bids.
- Verifiability. The result of the auction can be checked.
- Type Flexibility. The protocol supports a variety of auction types. For example, first price or Vickrey auctions.
- Bid Privacy. The losing bids remain confidential, even after the auction has ended.
- Bidder Anonymity. Identities of losing bidders kept secret.
- Robustness. Any malicious behaviour by any of the auction participants cannot cause an incorrect auction result.

Using these desirable properties as a guide, a taxonomy of existing privacy preserving online auctions has been constructed and is shown in Table 2.1. All the auction protocols studied have implemented the first three items in the list, correctness, confidentiality, and fairness. The taxonomy examines what other properties an auction protocol includes as well as how they have been implemented. The taxonomy presents the auction protocols in chronological order starting with the original paper by Franklin and Reiter.

2.1 Price Flexibility

The majority of the studied auction protocols do not provide price flexibility [2, 5, 10, 25, 26, 34, 36, 39, 41, 42, 43, 48, 53, 54]. Instead these auction protocols have a pre-defined set of allowable bid prices for an auction. These prices do not have to be linearly increasing and can provide higher bid precision for higher bids. It has been argued that five hundred bid values should be sufficient for any auction [34].

Auction protocols by Franklin and Reiter [16], Kikuchi [31], Cachin [7], and some of the models presented by Peng, Boyd, Dawson and Viswanathan [44] provide price flexibility which would provide a greater degree of flexibility to both the bidders and the auctioneers. An auction protocol that does not implement price flexibility would be able to emulate the behaviour of one that did simply by providing enough possible bid prices. For example, in an auction for a good that is expected to sell for one thousand dollars giving a bid range of one dollar to three thousand dollars provides a similar degree of freedom to an auction with full price flexibility. The capability of an auction protocol to provide a large number of possible bids and the performance cost of doing so seem to be the critical factors.

2.2 Verifiability

2.2.1 Group Verifiability

Group verifiability allows parties that were taking part in the auction to verify the auction process. Group verification is a valuable tool that can give auction participants confidence in the auction result.

The garbled circuit auction protocol by Naor, Pinkas, and Sumner [36] provides group verification. Bidders are able to verify that the auctioneer computed

Auction Protocol	Price Flex.	Verifiable		Support Combin.	Bid Privacy							Bid Anon.
					Trust Model			Level				
		Grp	Pub		Single	Thresh	2 Pty	0	1	s	*	
Franklin [16] + Reiter	X	X				X					X	
HKT [26]						X		X				
Cachin [7] 2 Server	X	X					X		X			
Garbled [36, 28] Circuits		X					X	X				
Non [2] Interactive					X			X				
Polynomial [31] Protocol	X	X				X		X				
No [34] Thresh. Trust		X					X	X				
Ext. Poly [48] Protocol				X		X		X				
Homo. [53] Protocol				X		X		X				
Extended [43] HKT			X			X		X				X
Five [44] Models	X		X			X				X		X
SGVA [54] Auction				X		X		X				
Yet [25] Another			X			X		X				
Receipt [10] Free			X				X	X				
Comb. [39, 38] Bidder Res.				X		X		X				
GW [42] Micali			X			X		X				
Verifiable [41] Protocol			X		X						X	
Bidder [5] Resolved			X			X		X				

Table 2.1: Secure Auction Taxonomy

the circuit given to it by the auction issuer, as well as verifying that their bid was counted in the auction. The auctioneer is also able to verify that the correct circuit was sent to it by the auction issuer. Group verification is provided using a variety of techniques including cut and choose verification (discussed in Section 3.3.1).

Lipmaa, Asokan, and Niemi have developed a two party trust auction model that provides similar group verification to garbled circuits [34]. The auctioneers can verify that the bidders submitted correct bids, and the auctioneer can verify the auction authority correctly computed the result to the auction. The group verification for this protocol is provided using range proofs.

Finally Kikuchi has developed an auction protocol that uses verifiable secret sharing to provide group verification [31]. Any participant can verify that the auctioneers have correctly computed the auction result, and the auctioneers can confirm that the bidders have submitted correct bids.

2.2.2 Public Verifiability

Public verifiability allows any third party to verify the auction process regardless of whether they were taking part in the auction. Although the main focus of verification is to give auction participants confidence in the auction result, public verification could be used by a reputation service that verifies all the auctions and keeps a reputation score depending on the number of auctions that an auctioneer or bidder has been involved in that fail the verification.

Brandt's bidder resolved auction protocol is publicly verifiable [5] using zero knowledge proofs. Zero knowledge proofs are submitted with every action taken by the bidders to calculate the auction result so a malicious bidder sending incorrect values is caught with a high probability.

Parkes, Rabin, Shieber, and Thorpe use a single auction server and rely on the public verification of the auction process to assure that the auction server is not corrupt [41]. This allows them to reduce the amount of communication overhead of the protocol as the auction is executed entirely on one host.

It is interesting to note that group verification was used in the earlier papers but on the later papers public verification is used.

2.3 Type Flexibility

Several different auction types can be used to conduct an electronic auction. Some of the more popular auction types are:

- First price auction. One good is available. The winning bidder bids the highest amount for the good, and they pay the price they bid.
- Vickrey auction. One good is available. The winning bidder bids the highest amount for the good, and they pay the price of the second highest bid.
- (M+1)st price auction. M goods are available. The M winning bidder bid the M highest amounts for the good, and they pay the price of the M+1 highest bid. A Vickrey auction is a special type of this auction where M=1.
- Combinatorial auction. N goods are available and bidders can bid on any combination of the goods. The goods are allocated so as to maximise the total price.

Some auction schemes support a variety of different auction types. For example, the homomorphic auction protocol [53] can support first price, Vickrey, or combinatorial auctions. Others, like Brandt's bidder resolved auction, only support first price or the Vickrey auction [5]. Two of the auction protocols have been extended to support combinatorial auctions. Brandt's bidder resolved auction protocol has been extended to support combinatorial auctions by Nzouonta [39, 38] and Kikuchi's auction protocol was also extended to support combinatorial auctions by Suzuki and Yokoo [48]. The original auction protocols both had verification that was lost when they were extended.

2.4 Bid Privacy Trust Model

To provide privacy in auctions, bids are encrypted or obfuscated in some way so that losing bid values are kept secret. The trust model used by an auction protocol is normally one of three basic trust models:

- Single Trusted Server.
- Threshold Trust.
- Two Party Trust.

2.4.1 Single Trusted Server

The auction protocol by Parkes, Rabin, Shieber and Thorpe uses a single trusted server that learns all the bid values after the bid submission phase [41]. The auction protocol concentrates on providing a public verification protocol so that even if the auctioneer is malicious, bidders will detect if the auctioneer has incorrectly calculated the auction result. Private information revelation is not prevented in this auction protocol.

An auction protocol by Baudron and Stern also uses a single server but this server is prevented from learning the losing bid values as long as it does not collude with one of the bidders [2]. A malicious auctioneer in this protocol could also sign up as a bidder and then be able to collude with itself and discover the losing bid values.

Using a single server is practical as only one auctioneer needs to be found to host the auction, but it does not provide robustness if the auctioneer fails in the middle of the auction thus is not often employed.

2.4.2 Threshold Trust

In the threshold trust model, trust is shared among a set of hosts. Unless a certain number (a quorum) of hosts are corrupt, the privacy of the protocol is preserved. Threshold trust has been used in a large number of auction protocols [16, 5, 31, 39, 48, 53, 54].

The threshold trust model can be further broken down in to (t, n) threshold schemes and (n, n) schemes. In (t, n) schemes, a number t out of n total servers would need to be corrupt to compromise bid privacy. The threshold value t is normally set to $n/2 + 1$. Common examples of values for (t, n) would be $(2, 3)$ or $(3, 5)$. In (n, n) schemes all n hosts need to be corrupt to compromise bid privacy.

The (t, n) threshold model is used in the majority of auction protocols [16, 31, 39, 48, 53, 54] to distribute the trust among a group of n auctioneers. Although it cannot be ruled out that t of the auctioneers are corrupt, the chances of them being corrupt can be reduced by careful selection of the auctioneers. The auctioneers should all be picked from different organisations, and should have different attributes such as operating systems where possible.

Brandt's bidder resolved auction model uses a (n, n) threshold trust model where the trust is shared between the bidders [5]. This means that unless all the bidders are corrupt bid privacy is preserved. If all bidders are corrupt it is impossible to preserve bid privacy as the bidders can share their private information amongst themselves. This model provides greater security than the (t, n) model, the bid privacy is preserved if even one of the bidders is honest.

Auction protocols using the (t, n) threshold trust model can achieve greater robustness than protocols using either the (n, n) threshold trust, a single server, or two party trust as they can tolerate up to $n - t$ corrupt or failed auctioneers and still be able to complete the auction. If an auction participant fails for some reason in an auction protocol using a (n, n) threshold trust model, the auction would need to restart because an (n, n) threshold trust model requires all n auction participants to decrypt the result of the auction.

2.4.3 Two Party Trust

In the two party trust model, trust is distributed between two separate parties. Unless these two parties collude, bid privacy is preserved. The two parties should be hosted by different organisations.

In the auction protocol of Naor, Pinkas, and Sumner the trust is distributed between the auctioneer and an auction issuer [36]. The privacy of the bids is preserved as long as these two parties do not collude. The auction issuer would be a well known organisation, where as the auctioneer would not necessarily be trusted by the bidders. This method allows for large numbers of untrusted auctioneers to run auctions generated by a small number of well known auction issuers. This is similar to the two party trust model used in an auction protocol developed by Lipmaa, Asokan, and Niemi [34].

Another two party trust auction protocol uses two auction servers, where together the two auction servers create a semi ordered list of encrypted bids which allows the identification of the highest bid [7]. The privacy of the bids is preserved as long as the two auction servers do not collude and as long as one of the auction servers does not collude with a bidder.

It is easier to find two separate servers from separate organisations when using two party trust than it would be to find the n different servers needed to use the threshold trust model. For this reason two party trust can be more practical than the threshold trust model but does not have the same robustness as the auction cannot complete if one of the two servers fails.

2.5 Bid Privacy Level

When bid privacy is implemented, it can provide different levels of privacy. These levels of privacy can be grouped as follows:

- Level 0. No information except the winning bidder and the price they paid are revealed.
- Level 1. Besides the information leakage of level 0, one other piece of information is revealed. For example, apart from the information revealed by level 0, the fourth highest bid could also be revealed.
- Level s. Besides the information leakage of level 0, it is also possible to recover bid statistics. For example, the maximum bid, the average bid, and the standard deviation of bids.
- Level *. All of the bids are revealed to the auctioneer after the auction closes.

A bid privacy level of * as provided by two of the auction protocols [16, 41] prevents insider trading as the auctioneer does not learn the the bid values until the bid submission phase has finished, but it does not prevent private information revelation as after the bid submission phase all the bids are revealed.

Both bid privacy level s and 1 prevent insider trading and may prevent private information leakage depending on the statistics or piece of information being revealed. Only the suggested auction protocols by Peng, Boyd, Dawson and Viswanathan [44] provide bid privacy level s. An auction protocol developed by Cachin [7] provides bid privacy level 1 as it leaks a partial ordering of the bids when the two auction servers are computing the result of the auction, however insider trading is prevented as this partial ordering is only learned after the bid submission phase has ended.

Bid privacy level 0 is the ideal case and is provided by most of the auction protocols studied. A bid privacy level of 0 prevents insider trading as well as private information revelation.

2.6 Bidder Anonymity

Two auction protocols developed by Peng, Boyd, Dawson and Viswanathan [44, 43] implement bidder anonymity. Bidder anonymity is achieved by adding an additional stage to the auction protocol during registration where bidders are provided with anonymous ids to sign bids with. This stage could be added to the other auction protocols mentioned above.

If both bidder anonymity and bid privacy are implemented, then no information can be learned about losing bidders, not even who was taking part, unless either the anonymity service or auction protocol is compromised. One of the biggest problems with providing bidder anonymity is that if there is no way to trace a bid to a bidder, then the winning bidder can repudiate on their bid. So bidder anonymity needs to provide some way to link the winning bidder to their bid if they do not acknowledge it is their bid, as well as stopping losing bidders from claiming to have submitted the highest bid.

A group signature scheme has been suggested as a method to provide bidder anonymity [51]. This is generated by two servers so that neither of the servers knows the true link between bidder and signature. In the case of a bid needing to be mapped to it's real bidder, the two servers can co-operate to find the true identity of the bidder from the bid's signature.

Another method to provide bidder anonymity is to use use blind signatures and a registration authority [44]. Bidders are issued a blind signature from the registration authority, and can generate a pseudonym from this signature. If the winning bidder does not identify themselves, all other bidders can prove using a 1 out of n zero knowledge proof that the winning bid is not signed by them, which will identify the winning bidder. One disadvantage of this scheme is that if one other bidder refuses to take part in the 1 out of n proof, the winning bidder cannot be identified.

For bidder anonymity to be effective, the bids must also be submitted through an anonymous communication channel like a mix-net to prevent the IP address a bid came from identifying the bidder.

2.7 Discussion

The goal of this thesis is to produce a privacy preserving, verifiable, combinatorial auction protocol with a bid privacy level of 0. Verification gives confidence in the result of the auction and prevents bid filtering and the malicious auctioneer. Support for combinatorial auctions provides a method to conduct the combinatorial auction examples provided in Chapter 1. Bid privacy of level 0 prevents both insider trading and private information revelation. None of the auction protocols studied provides all these properties, but the current auction protocols can be extended to provide them. The rest of this thesis examines two possible alternatives, extending a privacy preserving combinatorial auction to provide verification and extending a verifiable privacy preserving auction protocol to conduct combinatorial auctions. The main body of work in this thesis is extending the combinatorial auction protocol by Suzuki and Yokoo [53] to be verifiable. Adding verification to this auction protocol makes it verifiable and capable of computing combinatorial auctions with a bid privacy level of 0 using the threshold trust model. Previous work in this research group comparing auction protocols has also shown this auction protocol to have better performance than the polynomial auction protocol [6]. Another major component of this thesis is the extension of the garbled circuits auction protocol to compute combinatorial auctions. This extension will mean that the garbled circuit auction protocol is verifiable and able to conduct combinatorial auctions with a bid privacy level of 0 using two party trust. The garbled circuit auction protocol was chosen as it is an interesting contrast to the Suzuki and Yokoo auction protocol as it employs two party trust as opposed to threshold trust and uses a bit representation of bids as opposed to the bid vector notation.

Chapter 3

Background

Before presenting the verification protocols for the homomorphic auction protocol, details of the zero knowledge proofs used to build the verification protocols are presented as well as the details of the homomorphic auction protocol.

3.1 Zero Knowledge Proofs

Zero knowledge proofs were first introduced by Goldwasser, Micali, and Rackoff [22] and are used to prove the validity of some assertion, without revealing any information other than the validity of the assertion. A zero knowledge proof is a randomised protocol for two parties, a verifier and prover, in which the prover wishes to convince the verifier of the validity of a given assertion [19]. A zero knowledge proof typically consists of a commitment from the prover, a challenge from the verifier, and a response from the prover. They have been used in many areas including smart cards [45], eVoting [11, 24], and electronic auctions [5].

Figure 3.1 illustrates a famous zero knowledge proof known as Ali Baba's cave. The prover Alice wants to convince the verifier Bob that she knows the secret password to open a locked door between R and S. To prove this while not revealing the secret password used, Alice and Bob conduct the following steps:

- 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) without him knowing.

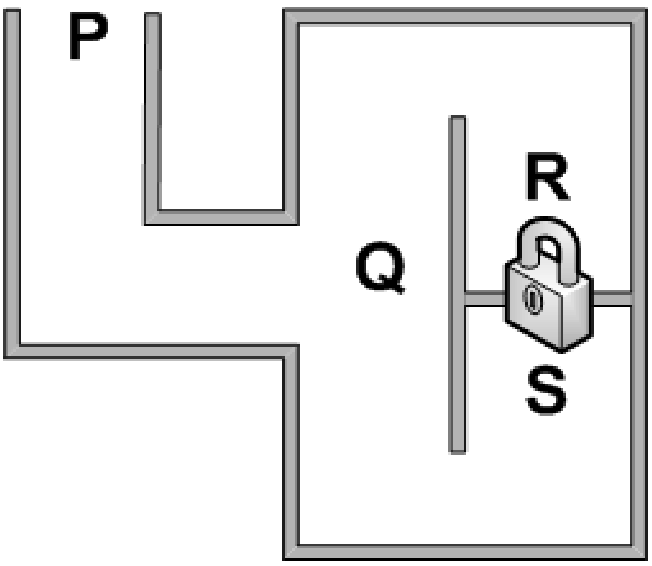


Figure 3.1: Ali Baba's Cave

- Bob chooses either the top (R) or bottom (S) tunnel.
- Bob challenges Alice to come out of the tunnel of his choice (challenge). Alice can only exit the correct tunnel 100% of the time if she knows the password.
- If Alice does not know the secret words, there is a 50% chance she will come out from the wrong tunnel (response).
- Bob can then repeat 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.

Ali Baba's cave provides a good illustration of a zero knowledge proof. A zero knowledge proof should have the properties of completeness, soundness, and zero knowledge. Completeness is the probability that the proof will succeed if the assertion being proved is valid. Soundness is the probability that the proof will succeed if the assertion is not valid. Zero knowledge is the property of not revealing any information other than the validity of the assertion being proved. Ideally zero knowledge proofs have 100% completeness, 0% soundness, and perfect zero knowledge. Most proofs do not meet all these requirements, there is normally a certain level of soundness greater than 0% which can often be improved by repeating the proof multiple times. Zero knowledge proofs are not a proof in the mathematical sense, instead they are constructed in such a way that it is infeasible for a cheating prover to publish a zero knowledge proof of an invalid assertion.

The zero knowledge proof in Figure 3.1 has a completeness of 100% because if Alice knows the secret password, she will always be able to pass the verification. With a single execution of this proof the soundness is 50% as there is a 50% she chose the right tunnel to go down and does not need the secret password. If the proof is executed n times, the soundness is $1/2n$.

A zero knowledge proof should yield nothing beyond the validity of the assertion being proved. So anything that is feasibly computable from the proof is also feasibly computable from the valid assertion. This property for zero knowledge is formalised using the simulation paradigm. Informally, the simulation paradigm states that an interactive proof A is zero knowledge on input x if for every feasible verifier strategy there exists a simulator S that on input x has output that is indistinguishable from the output of A. The proof A is then zero knowledge as the simulator S that is indistinguishable from A has no knowledge of anything other than the input x and the validity of the assertion.

For the zero knowledge proofs presented in this thesis the completeness and soundness of the proofs are examined. Simulators are also constructed to show zero knowledge. All the mathematical operations in the next section are computed modulo p in the group of integers $\mathbb{Z}_p$ closed under multiplication unless otherwise stated. The notation of $E(M) = (A = g^r, B = y^r M)$ is used for the homomorphic El-Gamal encryption, and $B/A^x = M$ for the decryption where g is a generator for the group $\mathbb{Z}_p$, q is some prime where $q|p-1$ (q divides $p-1$), x is the secret key, and $y = g^x$ is the public key for El-Gamal encryption.

3.2 Non-Interactive Zero Knowledge

Interactive zero knowledge proofs require the prover to answer challenges sent by the verifier to convince the verifier of the validity of the assertion being proved. In our setting of auctions this is a disadvantage as it could require provers to interact with verifiers for an indefinite time after the auction has been completed. A participant in the auction protocol could become disconnected from the network after the close of the auction or be busy conducting more auctions and be unable to take part in the interactive verification.

Non-interactive zero knowledge proofs [4] involve a prover publishing a proof that can be verified without interaction with the prover. The use of non-interactive proofs frees up provers to either disconnect from the network or compute other tasks after the close of the auction while still allowing verifiers to check the result of the auction. Non-interactive zero knowledge proofs make use of a random string or hash function available to both the prover and the verifier. In the implementation of the verification protocol, the cryptographic hash function SHA-512 is used as the hash function or random oracle for the zero knowledge proofs. The SHA-512 hash function has an output space of 512 bits. When constructing simulators for non-interactive zero knowledge proofs, it can be assumed that the simulator can define the output of the random oracle for a certain input[3][24]. When constructing zero knowledge proofs using a random oracle they are secure in the random oracle model.

The following proofs are used as building blocks for our verification protocols. They are well known proofs that have been altered for this thesis from being interactive, how they are normally presented, to being non-interactive using ideas from Damgard [13] and Goldreich [19]. The completeness, soundness, and zero knowledge of the proofs are presented in appendix A.

3.2.1 Proof of Knowledge of a Discrete Logarithm

The following is a non-interactive proof of knowledge of a discrete logarithm. The prover and verifier both know v and g, but only the prover knows x such that $v = g^x$, which is based on Schnorr's Σ-protocol [45].

The prover, P, conducts the following steps using hash function H:

- P picks a random number $z \in_R \mathbb{Z}_q$.
- P computes $a = g^z$.
- P computes $c = H(a)$.
- P computes $r = (z + cx) \bmod q$.
- P then publicly publishes the proof transcript a and r.

The verifier, V, conducts the following steps using the same hash function H and the proof transcript a and r:

- V computes $c = H(a)$.

- V accepts if $g^r = av^c$.

An Example - Proving an encrypted item decrypts to some value

An auctioneer may need to prove that an encrypted item decrypts to Z, but will not want to reveal it's secret key as that would enable the verifier to decrypt any other values encrypted. If the item decrypts to Z, $B/A^x = Z$ and therefore $B/Z = A^x$. Now it is proved that the auctioneer knows x such that $B/Z = A^x$ where the values (A, B) and Z are known to both the auctioneer and the verifier.

This example uses the encryption parameters $p = 23, q = 11, x = 2, y = 4, g = 2$, and $Z = 4$. Now when $r = 7$, $E(Z) = (2^7, 4^7 4) = (13, 9)$ and $B/Z = 9/4 = 9 * 6 = 8$. Now the auctioneer can prove that $B/Z = A^x$.

The auctioneer, or prover P, conducts the following steps using hash function H:

- P picks a random number $z = 13$.
- P computes $a = g^z = 13^{13} = 8$.
- P computes $c = H(a) = 6$.
- P computes $r = (z + cx) \bmod q = (13 + 2 * 6) = 25 \bmod q = 3$.
- P then publishes the proof transcript a and r.

The verifier, V, conducts the following steps using the same hash function H and the proof transcript a and r:

- V computes $c = H(a) = 6$.
- V accepts if $g^r = av^c$ where, in this example, $v = B/Z$. So V1 calculates $g^r = 13^3 = 12$ and $av^c = 8 * 8^6 = 12$ and checks they are equal.

3.2.2 Proof of Equality of Discrete Logarithms

The following is a non-interactive proof of equality of two discrete logarithms. The prover and verifier both know v, w, g_1 and g_2, but only the prover knows x so that $v = g_1^x$ and $w = g_2^x$. It is based on a protocol from Chaum and Pederson [9]. This zero knowledge proof is used to conduct verifiable threshold El-Gamal decryption in Section 5.3.1, and to prove the bid vectors are valid in Section 5.4 and in Section 6.3.

The prover, P, conducts the following steps using hash function H:

- P picks a random number $z \in_R \mathbb{Z}_q$.
- P computes $a = g_1^z$ and $b = g_2^z$.
- P computes $c = H(a + b)$.
- P computes $r = (z + cx) \bmod q$.
- P publishes the proof transcript a, b, and r.

The verifier, V, conducts the following steps using the same hash function H and the proof transcript a, b, and r:

- V computes $c = H(a + b)$.
- V checks that $g_1^r = av^c$ and $g_2^r = bw^c$.

An Example - Raising an encrypted value to a Random Exponent

If an auctioneer raises a publicly known El Gamal encrypted value to a random exponent, this proof can be used to prove the auctioneer knows the random value used without revealing it. The original encryption of a value is $E(M) = (A, B)$ and when it is raised to an exponent e it becomes $E(M)^e = (A^e, B^e) = (A_{exp}, B_{exp})$. The original values A, B and the results A_{exp}, B_{exp} are known but only the auctioneer knows e such that $A_{exp} = A^e$ and $B_{exp} = B^e$.

This example uses the encryption parameters to $p = 23, q = 11, x = 2, y = 4, g = 2$, and $z = 4$. Now when $r = 7$, $E(z) = (2^7, 4^7 4) = (13, 9)$. Now, say the auctioneer raises the encryption to power of $e = 9$. Then $(13, 9)^9 = (13^9, 9^9) = (3, 2)$. Now the auctioneer wishes to prove that it knows e without revealing it's value.

The auctioneer, or prover P, conducts the following steps using hash function H:

- P picks a random number $z = 15$.
- P computes $a = g_1^z = 13^{15} = 18$ and $b = g_2^z = 9^{15} = 6$ and publishes them on the bulletin board.
- P computes $c = H(a + b) = 21$.

- P computes $r = (z+cx) \bmod q = (15+21*9) = 204 \bmod q = 6$ and publishes it on the bulletin board.
- P publishes the proof transcript a, b, and r.

The verifier, V, conducts the following steps using the same hash function H and the proof transcript a, b, and r:

- V computes $c = H(a + b) = 21$.
- V checks that $g_1^r = av^c$ and $g_2^r = bw^c$. So V calculates $g_1^r = 13^6 = 6$ and $av^c = 18 * 3^{21} = 6$ and checks they are equal. V then calculates $g_2^r = 9^6 = 3$ and $bw^c = 6 * 2^{21} = 3$ and checks they are equal.

3.2.3 Proof an Encrypted Item Decrypts to 1 or Z

The following is a proof that an encrypted value $E(M) = (A = g^r, B = y^r M)$ either decrypts to a 1 or Z. It is based on a protocol from Cramer et al. [11]. This zero knowledge proof is used to prove the bid vectors are valid in Section 5.4 and in Section 6.3.

The prover, P, conducts the following steps using hash function H:

- If $M = 1$ P chooses r_1, d_1, w at random from $\mathbb{Z}_q$ and calculates $a_1 = g^{r_1} A^{d_1}$, $b_1 = y^{r_1}(B/Z)^{d_1}$, $a_2 = g^w$ and $b_2 = y^w$. If $M = Z$ P chooses r_2, d_2, w at random from $\mathbb{Z}_q$ and calculates $a_1 = g^w$, $b_1 = y^w$, $a_2 = g^{r_2} A^{d_2}$ and $b_2 = y^{r_2} B^{d_2}$.
- P computes $c = H(a_1 + a_2 + b_1 + b_2)$.
- If $M = 1$ P calculates $d_2 = c - d_1 \bmod q$ and $r_2 = w - rd_2 \bmod q$. If $M = Z$ P calculates $d_1 = c_i - d_2 \bmod q$ and $r_1 = w - rd_1 \bmod q$.
- P publishes the proof transcript a_1, a_2, b_1, b_2, d_1, d_2, r_1, and r_2.

Now to verify that the encrypted value either decrypts to a 1 or Z, verify V conducts the following steps using the same hash function H, the encrypted value (A, B) and the proof transcript a_1, a_2, b_1, b_2, d_1, d_2, r_1, and r_2:

- V computes $c = H(a_1 + a_2 + b_1 + b_2)$.
- V checks that $c = d_1 + d_2 \bmod q$, $a_1 = g^{r_1} A^{d_1}$, $b_1 = y^{r_1}(B/Z)^{d_1}$, $a_2 = g^{r_2} A^{d_2}$, and $b_2 = y^{r_2} B^{d_2}$.

An Example - Verifying Correct Values in a Bid Vector

Suppose a bidder is submitting an encrypted bid vector. The number of possible prices are $k = 4$, the public keys are $p = 23, q = 11, g = 2, Z = 4, y = 4$, and the secret is $x = 2$. Suppose the bidder is bidding the second possible price, then the bid vector is, $bid = (E(Z), E(Z), E(1), E(1))$. If the bid vector is encrypted using $r = 1$, $r = 2$, $r = 3$, and $r = 4$ it equals $bid = ((2, 16), (4, 18), (8, 18), (16, 3))$. To prove the item is either a 1 or Z the bidder proves the following for every item in the bid vector which is proved here for the second item:

The prover, P, conducts the following steps using hash function H:

- $M = Z$ so P chooses $r_2 = 1, d_2 = 2, w = 3$ at random and computes $a_1 = 8$, $b_1 = 18$, $a_2 = 18$ and $b_2 = 8$.
- P computes $c = H(a_1 + a_2 + b_1 + b_2) = 3$.
- $M = Z$ so P computes $d_1 = 1, d_2 = 2, r_1 = 1, r_2 = 1$.
- P publishes the proof transcript a_1, a_2, b_1, b_2, d_1, d_2, r_1, and r_2.

Now to verify that the encrypted value either decrypts to a 1 or a Z, verify V conducts the following steps using the same hash function H, the encrypted value (A, B) and the proof transcript a_1, a_2, b_1, b_2, d_1, d_2, r_1, and r_2:

- V computes $c = H(a_1 + a_2 + b_1 + b_2) = 3$.
- V checks that $c = d_1 + d_2 \bmod q = 3, a_1 = g^{r_1}A^{d_1} = 8, b_1 = y^{r_1}(B/Z)^{d_1} = 18, a_2 = g^{r_2}A^{d_2} = 18, b_2 = y^{r_2}B^{d_2} = 8$.

3.2.4 Proof of Equality of Two Logarithm Lists

The following is a non-interactive proof of equality of two logarithm lists. The prover and verifier both know $n, v, w, g_1, ..., g_n$ and $m_1, ..., m_n$, but only the prover knows $x_1, ..., x_n$ so that $v = \prod_{i=1}^{n} g_i^{x_i}$ and $w = \prod_{i=1}^{n} m_i^{x_i}$. It is based on a protocol from Chaum and Pederson [9]. It is used as Proof 4 in the Furukawa and Sako proof of a valid shuffle of encrypted values [17] which is described in Section 3.2.5.

The prover, P, conducts the following steps using hash function H:

- P picks n random numbers $z_1, ..., z_n \in_R \mathbb{Z}_q$.

- P computes $a_1, ..., a_n$ where $a_i = g_i^{z_i}$ and $b_1, ..., b_n$ where $b_i = m_i^{z_i}$.
- P computes $c = H(a_1 + ... + a_n + b_1 + ... + b_n)$.
- P computes $r_1, ..., r_n$ where $r_i = (z_i + cx_i) \bmod q$.
- P publishes the proof transcript $a_1, ..., a_n$, $b_1, ..., b_n$ and $r_1, ..., r_n$.

The verifier, V, conducts the following steps using the same hash function H and the proof transcript $a_1, ..., a_n$, $b_1, ..., b_n$ and $r_1, ..., r_n$:

- V computes $c = H(a_1 + ... + a_n + b_1 + ... + b_n)$.
- V checks that $\prod_{i=1}^n g_i^{r_i} = v^c \prod_{i=1}^n a_i$ and $\prod_{i=1}^n m_i^{r_i} = w^c \prod_{i=1}^n b_i$.

An Example - Checking shuffled items use the same matrix and random integers for both A and B

If an auctioneer does a verifiable shuffle of bid vectors, it needs to prove that for each of the shuffled items the same random integer and permutation matrix were used. A shuffled value can be written as

$$(AShuffled_i, BShuffled_i) = (g^{r_i} \prod_{j=1}^{n} A_j^{M_{ji}}, y^{r_i} \prod_{j=1}^{n} B_j^{M_{ji}})$$

where r_i is the random integer used in the re-encryption of this item, and M_{ij} is a permutation matrix for the shuffle. The auctioneer needs to prove that for each $AShuffled_i$ and $BShuffled_i$ the same r_i and M_{ij} were used.

This can be proved for a two item bid vector by setting $v = AShuffled_i$, $w = BShuffled_i$, $n = 3$, $g_1 = g$, $g_2 = A_1$, $g_3 = A_2$, $m_1 = y$, $m_2 = B_1$, $m_3 = B_2$, $x_1 = r_i$, $x_2 = M_{1i}$, and $x_3 = M_{2i}$. Then let $(A_i, B_i) = ((8,3),(13,8))$ and $(AShuffled_i, BShuffled_i) = ((4,16),(18,8))$ where $r_i = 6, 3$, and

$$M = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Now the auctioneer wishes to prove that it knows r_1 and M_{ij} without revealing their values.

The auctioneer, A, conducts the following steps using hash function H:

- A picks random numbers $z_1 = 15$, $z_2 = 7$, and $z_3 = 4$.
- A computes $a_1 = g_1^{z_1} = 2^{15} = 16$, $a_2 = g_2^{z_2} = 8^7 = 12$, $a_3 = g_3^{z_3} = 13^4 = 18$.

- A then computes $b_1 = m_1^{z_1} = 4^{15} = 3$, $b_2 = m_2^{z_2} = 3^7 = 2$, $b_3 = m_3^{z_3} = 8^4 = 2$.
- A computes $c = H(a_1 + a_2 + a_3 + b_1 + b_2 + b_3) = 10$.
- A computes $r_1 = (z_1 + cx_1) \bmod q = (15 + 10 * 6) = 75 \bmod q = 9$.
- A computes $r_2 = (z_2 + cx_2) \bmod q = (7 + 10 * 0) = 7 \bmod q = 7$.
- A computes $r_3 = (z_3 + cx_3) \bmod q = (4 + 10 * 1) = 14 \bmod q = 3$.
- P publishes the proof transcript $a_1, ..., a_n$, $b_1, ..., b_n$ and $r_1, ..., r_n$.

The verifier, V, conducts the following steps using the same hash function H and the proof transcript $a_1, ..., a_n$, $b_1, ..., b_n$ and $r_1, ..., r_n$:

- V computes $c = H(a_1 + a_2 + a_3 + b_1 + b_2 + b_3) = 10$.
- V checks that $\prod_{i=1}^{n} g_i^{r_i} = v^c \prod_{i=1}^{n} a_i$. So V calculates $\prod_{i=1}^{n} g_i^{r_i} = 2^9 * 8^7 * 13^3 = 6 * 12 * 12 = 13$ and $v^c \prod_{i=1}^{n} a_i = 4^{10} * 16 * 12 * 18 = 13$ and checks they are equal.
- V checks that $\prod_{i=1}^{n} m_i^{r_i} = w^c \prod_{i=1}^{n} b_i$. So V calculates $\prod_{i=1}^{n} m_i^{r_i} = 4^9 * 3^7 * 8^3 = 13 * 2 * 6 = 18$ and $w^c \prod_{i=1}^{n} b_i = 16^{10} * 3 * 2 * 2 = 18$ and checks they are equal.

3.2.5 Publicly Verifiable Shuffle of Encrypted Values

Several methods have been suggested for verifying a shuffle of El-Gamal encrypted values. These include methods by Groth [23], Neff [37], and Furukawa and Sako [17]. This project uses the Furukawa and Sako [17] algorithm.

Both the Groth [23] and Neff [37] algorithms are based on the principle that a polynomial remains the same under a permutation of it's roots. Using this principle, they are able to prove that a correct shuffle has been done without revealing what that shuffle is.

The Furukawa and Sako [17] algorithm is an interactive verification algorithm that uses a permutation matrix to prove in zero knowledge that the shuffle was done correctly without revealing the actual permutation used. A permutation of shuffled values involves a set of initial encrypted values (A, B), a set of shuffled and randomised values $(AShuffled, BShuffled)$, and a permutation π. This permutation can then be represented as a matrix M_{ij} where $i, j = 1, ..., n$. The proof of the shuffle then involves proving:

- Proof 1: Given A_i and $AShuffled_i$, $AShuffled_i$ can be expressed as
$$AShuffled_i = g^{r_i} \prod_{j=1}^{n} A_j^{M_{ji}}$$
using integers $r_1, ..., r_n$ and a permutation matrix M_{ij} that satisfies the condition
$$\sum_{h=1}^{n} M_{hi} M_{hj} = \begin{cases} 1 \bmod q & \text{if } i = j \\ 0 \bmod q & \text{if } i \neq j \end{cases}$$
- Proof 2: Given A_i and $AShuffled_i$, $AShuffled_i$ can be expressed as
$$AShuffled_i = g^{r_i} \prod_{j=1}^{n} A_j^{M_{ji}}$$
using integers $r_1, ..., r_n$ and a permutation matrix M_{ij} that satisfies the condition
$$\sum_{h=1}^{n} M_{hi} M_{hj} M_{hk} = \begin{cases} 1 \bmod q & \text{if } i = j = k \\ 0 \bmod q & \text{otherwise} \end{cases}$$
- Proof 3: The integers $r_1, ..., r_n$ and the matrix M_{ij} used in Proof 1 and Proof 2 are the same.
- Proof 4: For each pair $(AShuffled_i, BShuffled_i)$ the same integers $r_1, ..., r_n$ and matrix M_{ij} have been used.

Proof 1 and Proof 2 are proved using a series of products. Proof 3 is proved by applying the same integers and matrix to an independent set of randomly chosen basis that will need to be chosen at the commencement of the auction and known to the auctioneer. Proof 4 is proved using the proof of equality of two logarithm lists from Section 3.2.4 to prove the equality of $AShuffled_i = g^{r_i} \prod_{j=1}^{n} A_j^{M_{ji}}$ and $BShuffled_i = y^{r_i} \prod_{j=1}^{n} B_j^{M_{ji}}$. The completeness, soundness, and zero knowledge of this proof are presented in the appendix A.5. This zero knowledge proof of a correct shuffle of encrypted values is used in the public verification protocol in Section 6.4.

3.3 Other Verification Techniques

3.3.1 Cut and Choose Verification

The Naor, Pinkas, and Sumner garbled circuit auction scheme uses a cut and choose verification of the garbled circuit sent by the auction issuer to the auctioneer [36]. This involves the auction issuer sending $n > 1$ copies of the garbled circuit to the auctioneer who opens $n - 1$ of the garbled circuits, asks the auction issuer to remove the garbling, and then verifies that the circuit does in fact compute the correct auction result. In this scenario the probability of an incorrect circuit being detected is $(n - 1)/n$.

A similar technique to this could possibly be used to verify a set of auctioneers. Say the bidders decide on a certain index, say n < m, and they run the auction m times with only the n-th run being the 'official' auction. Then the bidders could reveal their bid values for the m-1 auction runs that were not 'official' and verify that the auctioneers are correctly computing the auction result. Unfortunately, this scheme relies on none of the bidders colluding with the auctioneers and telling them which auction is the 'official' run.

A drawback of this verification process is that it can be quite expensive, using garbled circuits it will involve sending multiple garbled circuits to the auctioneer, with every garbled circuit potentially being quite large [36](the actual sizes of the circuits used to conduct combinatorial auctions are presented in Section 8.6).

3.3.2 Verifiable Secret Sharing

Verifiable secret sharing was first developed to allow entities receiving shares of a secret to verify that the dealer had given them a correct share of the secret, and to allow the verification that shares being used in the reconstruction of the secret were also correct.

Kikuchi makes use of verifiable secret sharing to allow bidders to verify auctioneers, as well as to allow auctioneers to verify the bidders [31]. The auctioneer can verify that the share of a bid they have been sent by a bidder is correct. The auctioneer then publishes the sum of all the shares of bids, and the bidders can verify that the auctioneer has correctly computed the sum of the shares.

Publicly verifiable secret sharing allows anyone to verify the shares of a secret. The main advantage of publicly verifiable secret sharing is that verification is not restricted to dealers or secret share holders, but can be carried out by any third party [46].

3.3.3 Range Proofs

Lipmaa has developed a large amount of work on range proofs and their applications to auctions and e-voting [33]. Lipmaa then extended this work and used it to create a secure sealed bid auction scheme [34]. The auction scheme represents bids using homomorphic encryption and range proofs are used to first prove that the bid is constructed correctly, that is $bid = B^i$ where $i <= V + 1$. V is set to the maximum possible bid.

This range proof is then extended to provide a method to allow one of the parties in the two party trust model to verify the result of the auction is correct. Given a coin-extractable doubly homomorphic encryption scheme (like the Damgard-Jurik cryptosystem) you can prove that a value X is the second largest amount from a set of encrypted values [34].

3.4 The Homomorphic Auction Protocol

This section presents a brief overview of the auction protocol by Suzuki and Yokoo [53] which is used as the homomorphic auction protocol in this thesis. This protocol is capable of computing the outcome of single good or combinatorial auctions while keeping losing bid values secret. The outcome can be computed by either a single auctioneer or a group of auctioneers using threshold encryption techniques. In this thesis the threshold version of the auction scheme is used exclusively. This is because the threshold scheme provides greater protection of the privacy of losing bid values by spreading the trust over a group of auctioneers rather than a single auctioneer. If a single auctioneer is used, it would be possible for that auctioneer to decrypt all the bid values if it was malicious. When using a threshold scheme, at least t parties in a (t, n) secret sharing scheme must be malicious to be able to decrypt losing bid values.

3.4.1 El-Gamal Encryption System

The homomorphic auction protocol makes use of the El-Gamal asymmetric homomorphic encryption system [18] to encrypt bids. Asymmetric encryption uses a public key for encryption and a private key for decryption. Any party can encrypt a message using the public key of the recipient but only the recipient who knows the private key can decrypt the message. Homomorphic encryption is a form of encryption where an algebraic operation can be performed on the plain-text of an encrypted item by performing an algebraic operation on the cipher-text. In the El-Gamal encryption scheme performing a multiplication of cipher-texts performs a multiplication of the underlying plain-texts.

The El-Gamal encryption scheme uses two large primes number p and q where $q|p-1$, the cyclic group of integers $\mathbb{Z}_p$ closed under multiplication, and a generator of the group g as public values known to all parties. A secret key x for a participant is chosen randomly from the range $0, ..., q-1$ and the public key is calculated as $y = g^x$. An El-Gamal encryption involves choosing a random value r and calculating the encryption of a value m as $E(m) = (g^r, y^r m) = (A, B)$. The decryption of $E(m)$ is computed by calculating $D(E(m)) = B/A^x = y^r m/g^r x = g^r x m/g^r x = m$.

An example of El-Gamal encryption is now given where $p = 23$, $q = 11$, $g = 2$, and $x = 4$ and all operations are done on the group Z_{23}. All operations are done modulo p unless otherwise stated. The public key is calculated by computing $y = g^x = 2^4 = 16$. Suppose the message $m = 6$ is being encrypted using random value $r = 7$. The encryption is calculated $E(m) = (g^r, y^r m) = (2^7, 16^7 * 6) = (13, 16)$. To decrypt this, calculate $B/A^x = 16/13^4 = 16/18 = 16 * 9 = 6$. Division is done by multiplying the dividend by the multiplicative inverse of the divisor. In this case the multiplicative inverse of $18 \bmod 23 = 9$.

Using El-Gamal encryption a new randomised ciphertext can be created by multiplying the original ciphertext by an encryption of 1. If the decisional Diffie-Hellman (DDH) problem is infeasible, one cannot determine whether a cipher-text is a randomised ciphertext of the original or not. Using the values from the above example, suppose there is an El-Gamal encrypted value of $M = 6$, $E(M) = (13, 16)$. Now 1 can be encrypted using the random value $r = 3$ and the same keys as above by computing $E(1) = (g^r, y^r M) = (2^3, 16^3 * 1) = (8, 2)$. If $E(M)$ multiplied by $E(1)$ it gives $Product = (13, 16)*(8, 2) = (12, 9)$. The value $Product$ can be decrypted by calculating $D(Product) = B/A^x = 9/12^4 = 9/13 = 9*-7 = 6$. $E(M)$ and $Product$ look like totally different cipher-texts and it is not known whether the original plain-texts are the same without decrypting them. This randomisation technique is used in the shift and randomise step of the auction protocol described in Section 3.4.5.

3.4.2 Auction Graphs

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

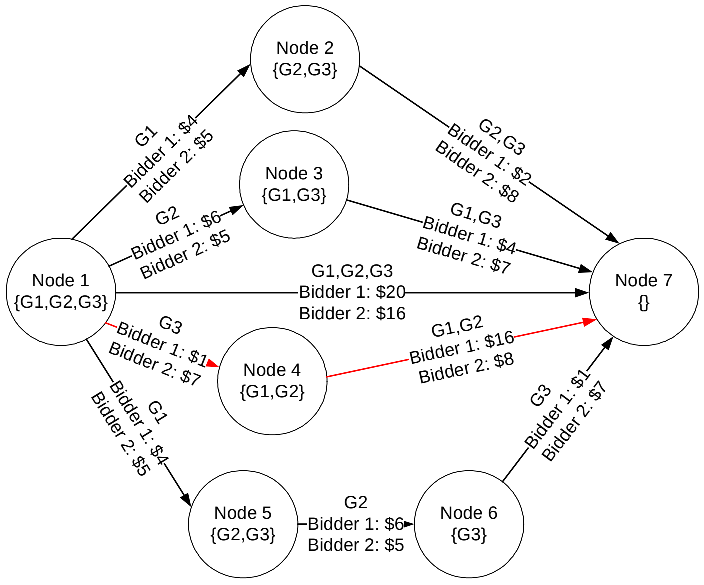

Node 1 {G1,G2,G3}
Node 2 {G2,G3}
Node 3 {G1,G3}
Node 4 {G1,G2}
Node 5 {G2,G3}
Node 6 {G3}
Node 7 {}
G1 Bidder 1: $4 Bidder 2: $5
G2 Bidder 1: $6 Bidder 2: $5
G1,G2,G3 Bidder 1: $20 Bidder 2: $16
G3 Bidder 1: $1 Bidder 2: $7
G1 Bidder 1: $4 Bidder 2: $5
G2,G3 Bidder 1: $2 Bidder 2: $8
G1,G3 Bidder 1: $4 Bidder 2: $7
G1,G2 Bidder 1: $16 Bidder 2: $8
G3 Bidder 1: $1 Bidder 2: $7
G2 Bidder 1: $6 Bidder 2: $5

Figure 3.2: Example Auction Graph

Figure 3.2 provides an example auction graph for three goods $G1$, $G2$, $G3$ with two bidders. Each edge is labelled for the subset of goods it represents along with the bids of the two bidders for that allocation. A complete path through the graph is an allocation of goods and the auctioneer needs to find the optimal path through the graph to compute the auction. An optimal path is a complete path where there are no other paths that provide greater value. The optimal path is shown in this graph as the thinner red line. The optimal allocation for this auction is to allocate $G3$ to bidder 2 for $\$7$ and $G1, G2$ to bidder 1 for $\$16$ for a total of $\$23$. There may be more than one optimal path in a graph, but there should be no other paths that have a greater value.

3.4.3 Bid Vectors

Bids in the homomorphic auction protocol are represented by encrypted bid vectors. The bid vectors are composed of a series of encryptions of a publicly known value Z followed by encryptions of 1. The value of the bid vector is indicated by the number of Z values encrypted. For example, a bid of value 3 and length 6 is represented by the bid vector $E(Z), E(Z), E(Z), E(1), E(1), E(1)$ and a bid vector of value 2 and length 5 is $E(Z), E(Z), E(1), E(1), E(1)$. The bid vectors length must be at least as long as the maximum bid value for a complete allocation.

3.4.4 Finding the Maximum bid for a Node

To find the maximum bid value for a node while keeping losing bid values secret the auctioneers compute the product of all the bids for on the incoming edges for a node on the auction graph. This product is then decrypted item-wise from right to left until the first value that does not equal one is decrypted.

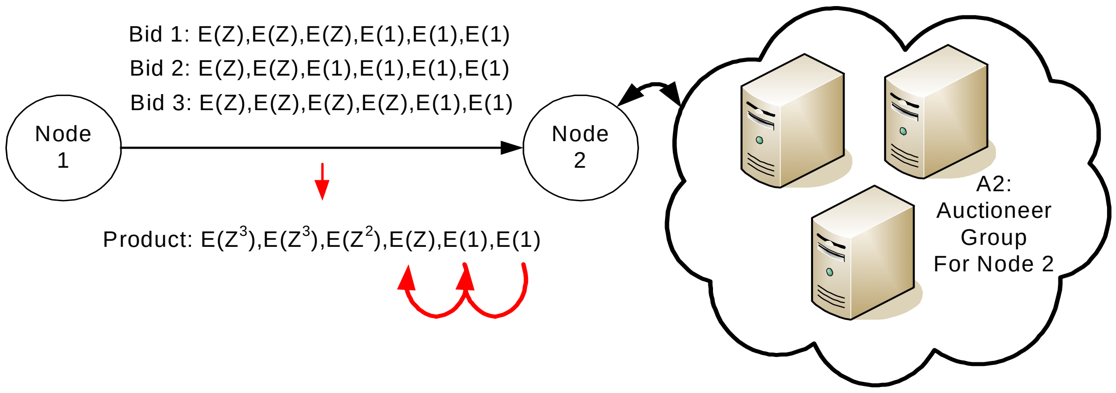

Bid 1: E(Z),E(Z),E(Z),E(1),E(1),E(1)
Bid 2: E(Z),E(Z),E(1),E(1),E(1),E(1)
Bid 3: E(Z),E(Z),E(Z),E(Z),E(1),E(1)
Node 1
Node 2
Product: E(Z3),E(Z3),E(Z2),E(Z),E(1),E(1)
A2: Auctioneer Group For Node 2

Figure 3.3: Finding the Maximum Bid for a Link

Figure 3.3 illustrates this process for the auctioneer group at Node 2 to find the maximum bid for the node. The auctioneers first compute the item-wise product of all the bid vectors on the incoming links. This can be done by any auctioneer due to the homomorphic nature of the encryption. The auctioneers then publish their shares of the decryption of the item on the far right of the product vector and use Lagrange interpolation to decrypt once t (from the (t, n) threshold encryption scheme) shares of the decryption have been published. They will then repeat this decryption from right to left until they find an item in the product vector that does not equal 1. In this example, the item in the fourth place of the product vector will decrypt to Z and so the auctioneers know that the maximum bid for this node is 4 and learn nothing about the values of the losing bids.

3.4.5 Shift and Randomise

Once the auctioneers have found the maximum bid m for the node, they shift and randomise any bid vectors on the outgoing links of the auction graph to add the maximum bid value to these bid vectors. The auctioneers first shift the bid vectors on the outgoing links right by m places and add m encryptions of Z which adds the value of m to the bids vectors. The auctioneers then create a bid vector composed entirely of encryptions of 1 and multiply the shifted vector by this randomising vector. Multiplying the items in the shifted bid vectors by encryptions of 1 randomises the vector so the value of m remains hidden and prevents other parties from counting the new items in the shifted vectors. Due to the homomorphic nature of the encryption, multiplying by encryptions of 1 preserves the value of the shifted bid vector.

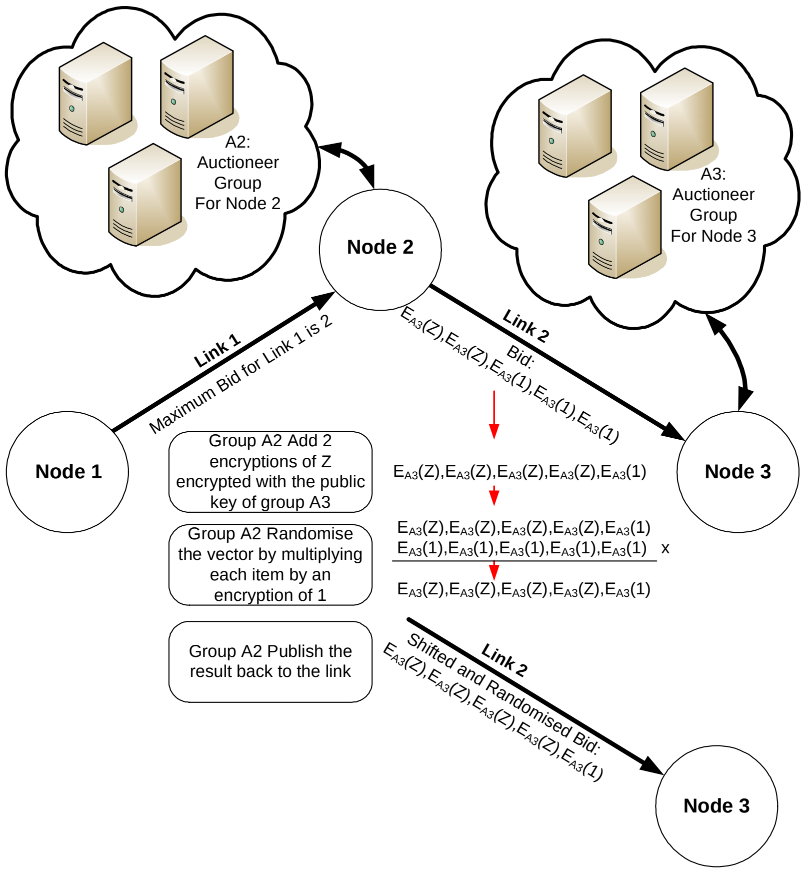

A2: Auctioneer Group For Node 2
A3: Auctioneer Group For Node 3
Node 2
Node 1
Node 3
Link 1
Maximum Bid for Link 1 is 2
Link 2
Bid: EA3(Z),EA3(Z),EA3(1),EA3(1),EA3(1)
Group A2 Add 2 encryptions of Z encrypted with the public key of group A3
EA3(Z),EA3(Z),EA3(Z),EA3(Z),EA3(1)
Group A2 Randomise the vector by multiplying each item by an encryption of 1
EA3(Z),EA3(Z),EA3(Z),EA3(Z),EA3(1)
EA3(1),EA3(1),EA3(1),EA3(1),EA3(1) x
EA3(Z),EA3(Z),EA3(Z),EA3(Z),EA3(1)
Group A2 Publish the result back to the link
Link 2
Shifted and Randomised Bid: EA3(Z),EA3(Z),EA3(Z),EA3(Z),EA3(1)
Node 3

Figure 3.4: Shifting and Randomising a Bid Vector

Figure 3.4 illustrates the process of shifting and randomising a bid vector. The auctioneers in group $A2$ have already computed the maximum bid for node 2 and now shift and randomise the bid on link 2 by the value $m = 2$. The auctioneers first shift the original bid vector right by 2 places and add 2 encryptions of Z. They then multiply the shifted bid vector by a vector of encryptions of 1 and publish it on link 2.

3.4.6 An Example

To calculate the result of an auction the auctioneers use the techniques in the earlier sections to find the optimal path for the auction graph. They then back track through the optimal path to find the optimal bids on the edges of the optimal path.

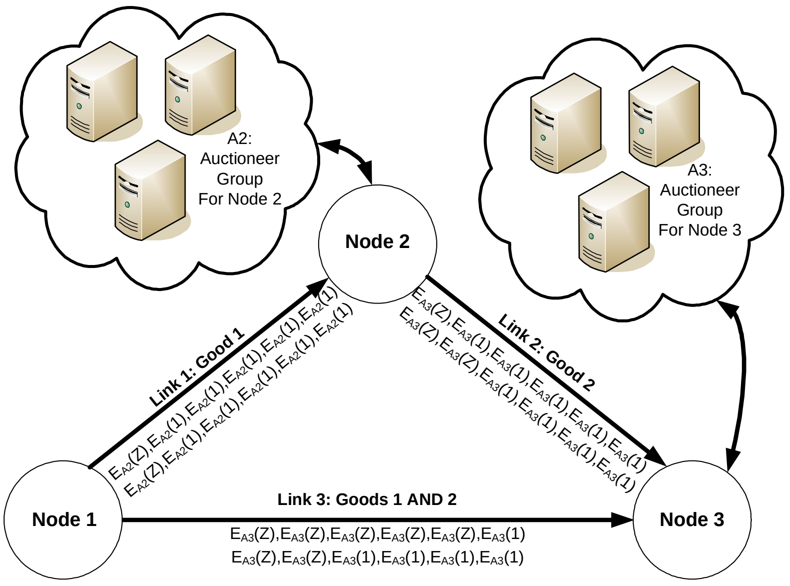

A2: Auctioneer Group For Node 2
A3: Auctioneer Group For Node 3
Node 1
Node 2
Node 3
Link 1: Good 1
$E_{A2}(Z),E_{A2}(1),E_{A2}(1),E_{A2}(1),E_{A2}(1),E_{A2}(1)$
$E_{A2}(Z),E_{A2}(1),E_{A2}(1),E_{A2}(1),E_{A2}(1),E_{A2}(1)$
Link 2: Good 2
$E_{A3}(Z),E_{A3}(1),E_{A3}(1),E_{A3}(1),E_{A3}(1),E_{A3}(1)$
$E_{A3}(Z),E_{A3}(Z),E_{A3}(1),E_{A3}(1),E_{A3}(1),E_{A3}(1)$
Link 3: Goods 1 AND 2
$E_{A3}(Z),E_{A3}(Z),E_{A3}(Z),E_{A3}(Z),E_{A3}(Z),E_{A3}(1)$
$E_{A3}(Z),E_{A3}(Z),E_{A3}(1),E_{A3}(1),E_{A3}(1),E_{A3}(1)$

Figure 3.5: A Simple Example Auction

Figure 3.5 shows an example auction graph with encrypted bid vectors published to the links of the graph. To compute the result of the auction the auctioneers will complete the following actions:

- The auctioneers in group $A2$ find the maximum bid for node 2 by multiplying the bids on the incoming links together and decrypting them from right to left until the first item is found that does not decrypt to 1. The optimal value for this node $optimal_2 = 1$.

- The auctioneers in group $A2$ shift and randomise the bid vectors for link 2 by the value $optimal_2$.

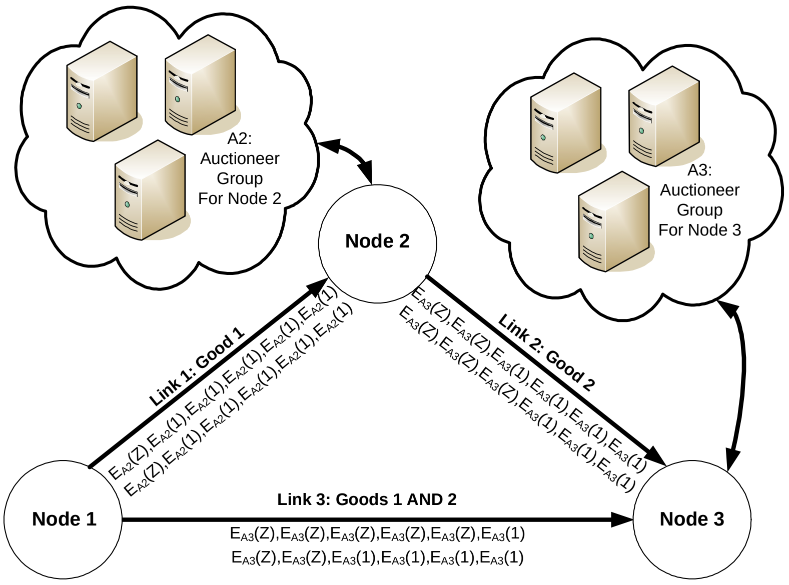

A2:
Auctioneer
Group
For Node 2
A3:
Auctioneer
Group
For Node 3
Node 1
Node 2
Node 3
Link 1: Good 1
E_A2(Z),E_A2(1),E_A2(1),E_A2(1),E_A2(1),E_A2(1)
E_A2(Z),E_A2(1),E_A2(1),E_A2(1),E_A2(1),E_A2(1)
Link 2: Good 2
E_A3(Z),E_A3(Z),E_A3(1),E_A3(1),E_A3(1),E_A3(1)
E_A3(Z),E_A3(Z),E_A3(Z),E_A3(1),E_A3(1),E_A3(1)
Link 3: Goods 1 AND 2
E_A3(Z),E_A3(Z),E_A3(Z),E_A3(Z),E_A3(Z),E_A3(1)
E_A3(Z),E_A3(Z),E_A3(1),E_A3(1),E_A3(1),E_A3(1)

Figure 3.6: After Shifting the Bids for Link 2

- Figure 3.6 shows the auction graph once the auctioneers in group $A2$ have shifted the bid vectors for link 2 by the value $optimal_2 = 1$.

- The auctioneers in group $A3$ now multiply the the bid vectors on the incoming links 2 and 3 together to form a product bid vector. The auctioneers in group $A3$ decrypt the items in the product bid vector from right to left until the first item that does not decrypt to 1 is found to find the optimal value for this node $optimal_3 = 4$. As this is the final node in the auction graph this is the optimal value for the auction.

- The auctioneers in group $A3$ now trace back the optimal value to find the winning bids. This is done by decrypting the bid vectors on the incoming links at the index $optimal_3$ to find the bid that decrypts to Z at this index. In this case the bid from bidder 1 on link 3 decrypts to Z at the index $optimal_3 = 4$. This means that the winning bid for the auction is bidder 1 on link 3 for goods 1 and 2 together.

3.5 Summary

This chapter has presented background work that is used in the rest of this thesis. It began by presenting the concepts of zero knowledge proofs and non-interactive zero knowledge proofs that are used to construct the group verification protocol in Chapter 5 and the public verification protocol in Chapter 6. A number of well known zero knowledge proofs have been presented. The zero knowledge proof of equality of discrete logarithms presented in Section 3.2.2 is used to conduct verifiable threshold El-Gamal decryption in Section 5.3.1 and to prove the bid vectors are valid in Section 5.4 and in Section 6.3. The zero knowledge proof an encrypted item decrypts to 1 or Z in Section 3.2.3 is used to prove the bid vectors are valid in Section 5.4 and in Section 6.3. The public verification protocol uses the zero knowledge proof of a correct shuffle of encrypted values from Section 3.2.5 to prove that a bid vector from a set encrypted bid vectors is the maximum bid in Section 6.4.

The rest of this chapter has detailed the homomorphic auction protocol by Suzuki and Yokoo [53]. The individual constructions and steps of the auction protocol are explained before presenting a complete example auction. This is important background work for this thesis as both the group verification protocol and the public verification protocol add verification to the homomorphic auction protocol and so an understanding of the auction protocol is necessary to construct and understand the verification protocols.

Chapter 4

Extending Garbled Circuits

Garbled circuits are a verifiable privacy preserving auction protocol first suggested by Naor, Pinkas, and Sumner [36]. They make use of the cryptographic technique of garbled circuits introduced by Yao [52]. Garbled circuits use a two party trust model with an Auction Issuer and an Auctioneer. As long as these two parties do not collude the auction is privacy preserving.

A contribution of this thesis is the construction of a novel circuit that can compute combinatorial auctions taking as inputs the number of bidders, the maximum price, and the number of goods. In previously published work, garbled circuits had only been used to compute (M+1)st price auctions. A circuit is a network of Boolean gates with a set of inputs, a set of intermediate gates, and a set of outputs gates. When the parameters of the function are presented bit wise to the input gates, the intermediate gates are used to compute the values of the output gates which are the result of the function the circuit is computing.

Figure 4.1 shows a circuit that is being used for a trivial auction with one good, two bidders and two bits representing the price. The input of the circuit is the bids for the two bidders represented by two bits. The outputs are two Boolean values that indicate whether bidder one or bidder two was the winner and two Boolean values that indicate the maximum or winning price. The example in Figure 4.1 shows bidder two winning the auction with a maximum price of 11 which is three. To further illustrate consider another example. Bidder one bids two so has input 10 and bidder two bids one and so has input 01. In this case, the output is Bidder 1 Winner = 1, Bidder 2 Winner = 0, Maximum Price Bit 1 = 1, and Maximum Price Bit 2 = 0.

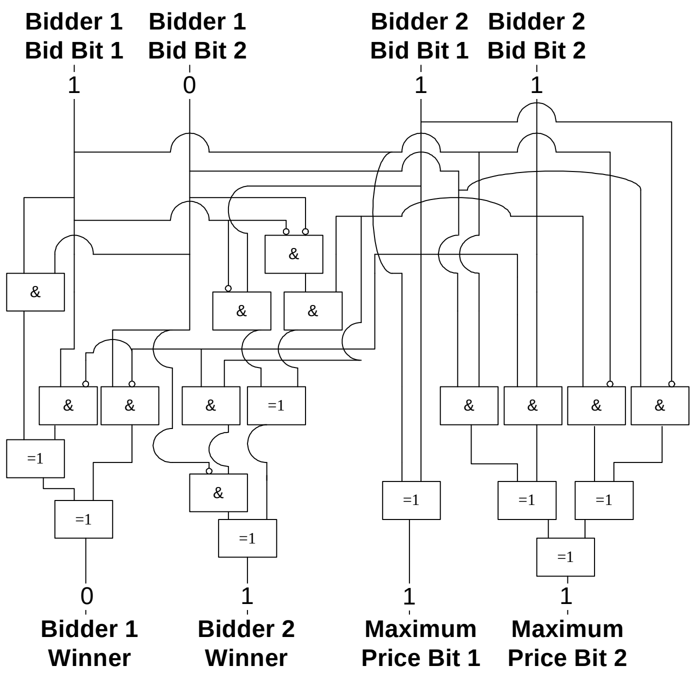


Figure 4.1: A Simple Auction Circuit

4.1 Garbled Circuit Auction Protocol

Figure 4.2 shows the parties involved in the garbled circuits auction protocol. The protocol preserves the communication pattern of non-electronic auctions so 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.

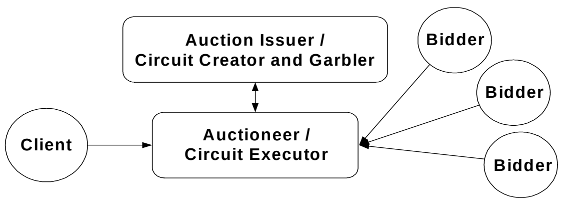

Auction Issuer / Circuit Creator and Garbler
Bidder
Bidder
Client
Auctioneer / Circuit Executor
Bidder

Figure 4.2: Garbled Circuit Parties

The basic steps of an auction using the garbled circuit protocol are:

- The client contacts the auctioneer with details of the auction they wish to run.

- The auctioneer advertises details of the auction including the number of goods, number of prices, and the auction issuer being used.
- The auction issuer constructs a garbled circuit for the auction based on how many bidders, goods, and the number of bits in the price as well as a mapping from the garbled outputs of the garbled circuit to the actual outputs of the auction and sends them to the auctioneer.
- The auction issuer, auctioneer, and bidders use a protocol called verifiable proxy oblivious transfer (VPOT) [28] 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.

4.2 Algorithms

4.2.1 Table of Definitions

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

- Client: The entity that requests the auctioneer to conduct an auction.
- Auctioneer: Takes the details from the client and runs the auction. Communicates with the auction issuer to get the garbled circuit and garbled input values.
- Auction Issuer: Assists in running the auction. Should be from a separate organisation than the auctioneer. Garbles circuits and then assists the auctioneer in learning the garbled inputs.
- Bidder: Bids on items in the auctions.
- Auction Circuit: Circuit composed of Boolean gates that can be used to compute the result of an auction.
- Node: Boolean gate in an auction circuit.
- Wire: Link between two nodes of an auction circuit. A wire can have a value b of 0 or 1.

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

4.2.2 Garbled Circuit Generation

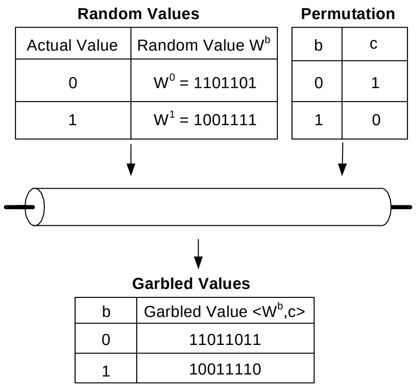
To garble a circuit, the auction issuer executes the following algorithm on the nodes and wires of the auction circuit.

Algorithm *GarbleCircuit*
Input: AuctionCircuit AC, RandomFunction F
Output: GateTable GT, OutputMapping OM

```
1.  (∗ Assign random values to the wires ∗)
2.  for ∀ wire $i \in$ AC
3.          Randomly generate $W_i^0$ and $W_i^1$ corresponding to 0 and 1.
4.          Choose a random permutation over {0,1}, $\pi_i : b_i \rightarrow c_i$.
5.  (∗ Construct function tables for every node ∗)
6.  for ∀ node $k \in$ AC with input nodes i,j
7.          for $c_i \leftarrow 0$ to 1
8.                  for $c_j \leftarrow 0$ to 1
9.                          $GT(k)(c_i, c_j) \leftarrow$
10.                         $GetGTValue(i, j, k)$
```

```
11. (* Construct output mapping *)
12. for ∀ output wire $k \in$ AC
13.         $OM(k,0) \leftarrow < W_k^0, \pi_k(0) >$
14.         $OM(k,1) \leftarrow < W_k^1, \pi_k(1) >$
```

Algorithm *GarbleCircuit* garbles an auction circuit. The first step is to assign random values to every wire of the auction circuit. Every wire has a random value corresponding to 0 and 1 (W^0, W^1) assigned to it as well as a random permutation of its output $\pi : b \rightarrow c$ that permutes the wires value b to c. Figure 4.3 shows the random values W^0 and W^1 assigned to a wire as well the permutation from $b \rightarrow c$. It shows the garbled values at the bottom of the image, showing how these are created by concatenating the random values and the permutation.



Random Values

Actual Value	Random Value W^b
0	W^0 = 1101101
1	W^1 = 1001111

Permutation

b	c
0	1
1	0

Garbled Values

b	Garbled Value $<W^b,c>$
0	11011011
1	10011110

Figure 4.3: Garbling a Wire

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

Algorithm *GetGTValue*
Input: InputNode i, InputNode j, OutputNode k
Output: bit [] Value

```
1. $Value \leftarrow$
2. $\{(W_k^{g(b_i,b_j)}, c_k) \oplus (F(W_i^{b_i}, c_j)) \oplus (F(W_j^{b_j}, c_i))\}$
```

4.2.3 Executing a Circuit

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

Algorithm *ExecuteCircuit*
Input: AuctionCircuit AC, GateTables GT, OutputMapping OM, GarbledInputs GI, RandomFunction F
Output: ActualValues AV

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

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

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

4.2.4 Worked Example

Figure 4.4 illustrates a trivial garbled circuit with an AND and an OR gate. Both the AND and the OR gate have one bit inputs and outputs. This circuit has three inputs and one output.

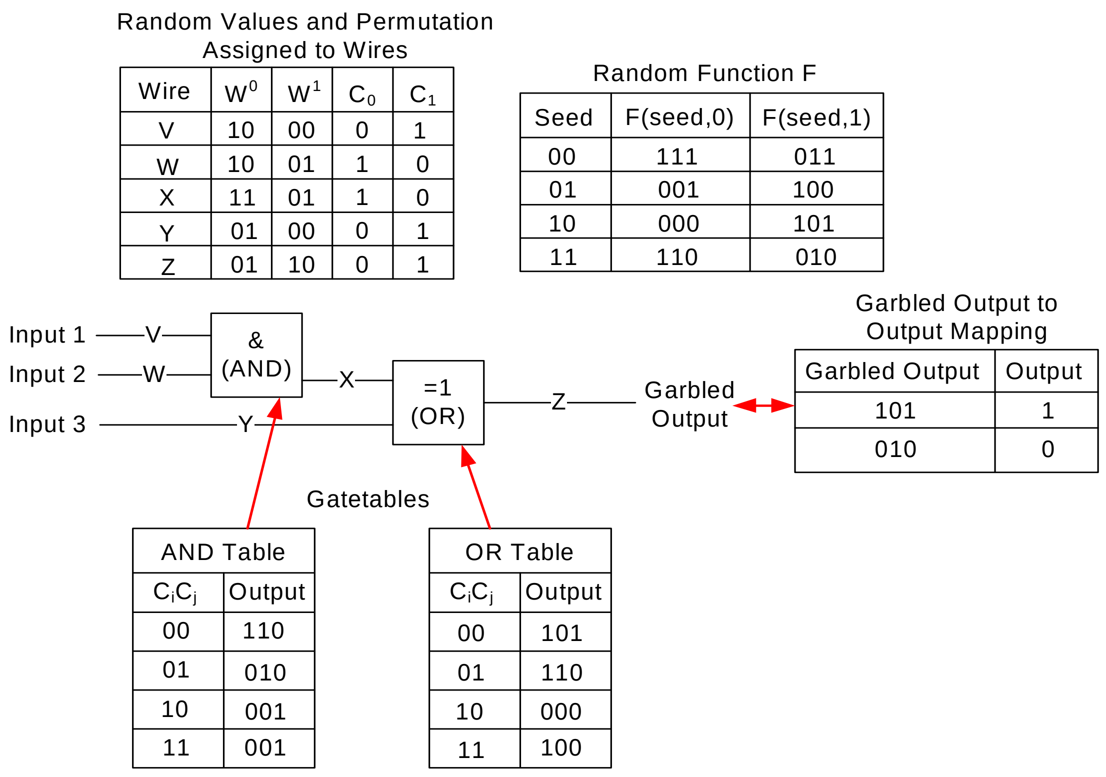
The auction issuer has executed the *GarbleCircuit* algorithm to produce the 'Random Values and Permutation Assigned to Wires' table shown in Figure 4.4 that contains the random values assigned to each wire in the example as well as the permutation. These random values are kept private by the auction issuer and not revealed either to the auctioneer or to the bidders. The auction issuer also produces the 'Gatetables' and the 'Garbled Output to Output Mapping' shown in Figure 4.4 using the *GarbleCircuit* algorithm. The garbled value of a wire is set to $< W^b, c >$ so for wire Z the garbled value of 0 is $< 01, 0 >= 010$ as shown in Figure 4.3. The 'Random Function F' shown in Figure 4.4 is a public function available to both the auctioneer and the auction issuer.

To execute the circuit in 4.4 the auctioneer would take the following steps:

- Find out the garbled input values. In this example we will set Input 1=1, Input 2=1, and Input 3=0. Given these example inputs, the output of the circuit should be 1. The garbled input value for V is 001, for W is 010, and for Y is 010. The garbled input value is the garbled value of the wire for the input value of the wire. So as Input 1 = 1, the garbled value for V is $< W^1, c_1 >= 001$.

- Now the gates are executed. To execute the AND gate the garbled inputs and the gatetable are used. The output is $001 \oplus 111 \oplus 100 = 010$.

- Now the OR gate is executed. The output is $101 \oplus 001 \oplus 001 = 101$. Using the garbled output to output mapping, the output of the garbled circuit is 1.



Random Values and Permutation Assigned to Wires

Wire	W^0	W^1	C_0	C_1
V	10	00	0	1
W	10	01	1	0
X	11	01	1	0
Y	01	00	0	1
Z	01	10	0	1

Random Function F

Seed	F(seed,0)	F(seed,1)
00	111	011
01	001	100
10	000	101
11	110	010

Garbled Output to Output Mapping

Garbled Output	Output
101	1
010	0

AND Table

C_iC_j	Output
00	110
01	010
10	001
11	001

OR Table

C_iC_j	Output
00	101
01	110
10	000
11	100

Figure 4.4: Garbled Circuit Example

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

4.3 Extending Garbled Circuits

The main contribution of this thesis to improving the garbled circuits auction protocol is an algorithm for the construction of an auction circuit that can compute the result of a combinatorial auction. While this is not a modification to the protocol itself, it is a construction that allows us to conduct combinatorial auctions using the garbled circuits protocol. Previously published work has used garbled circuits to conduct single good auctions only.

Auction circuits for the garbled circuit auction protocol need to be constructed dynamically based on the number of bidders, the maximum price, and the number of goods for the auction. A circuit for computing the result of an auction with ten bidders will have more gates than a circuit for conducting an auction with five bidders. An algorithm is needed that can return a combinatorial auction circuit for computing the result of an auction taking as inputs the number of bidders, the maximum price, and the number of goods.

The design and implementation of the combinatorial auction circuit began by looking at implementations of single good auction circuits. Kurosawa and Ogata [32] have constructed an algorithm for constructing auction circuits that compute first or second price single good auctions efficiently. These auction circuits take the bids from every bidder as input and output the winning bidder and the winning price for the auction. The auction circuit constructed is composed of NOT, AND, OR, XOR, and SELECT nodes. A SELECT gate has three inputs, if the first input is true it outputs the second input, and if the first input is false it outputs the third input. A SELECT gate is not a true Boolean gate, but it can be implemented using AND and OR gates and makes construction of circuits easier. The first price auction circuit is used as a building block for constructing the combinatorial circuit.

A basic add circuit was also constructed that given two bit-wise values as input, outputs the sum of these two values. This circuit was based on the Boolean gates used in a digital adder circuit for addition of binary values [30].

4.3.1 The Combinatorial Auction Circuit

To develop an algorithm to dynamically construct an auction circuit based on the number of bidders, goods, and the maximum price, the first step is to show how an auction circuit might be constructed that could compute the outcome of a combinatorial auction. Using the single good auction circuit and the add circuit, the actions in the homomorphic auction protocol can be computed. The single good circuit computes the maximum bid for a set of bids and the add circuit can be used to conduct the shift and randomise operation to add the maximum bid for a node to the bids on the outgoing links for that node.

The homomorphic auction protocol uses an auction graph to represent combinatorial auctions as shown in Figure 4.5 and described in Section 3.4.2. Using the auction graph and the circuits for computing the maximum bid for an auction and for adding two values together an auction circuit to compute the optimal value for a combinatorial auction was designed and implemented.

For every node in the auction graph the single good auction circuit is used to compute the maximum bid for the bids on the incoming links. Then the add

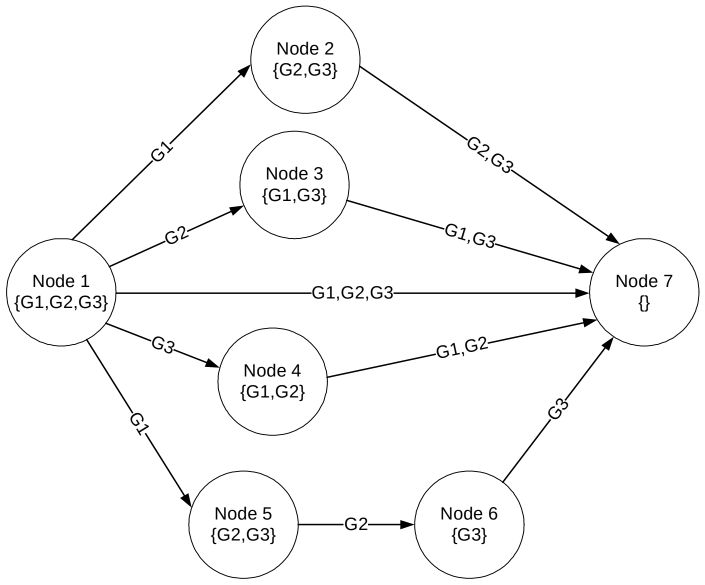


Figure 4.5: A Three Good Auction Graph

circuit is used to add this maximum value to all the bids on the outgoing link. Figure 4.6 shows the auction graph from Figure 4.5 with auction circuits in place to construct our combinatorial auction circuit. Every link has a maximum bid circuit attached to it that outputs the maximum bid for that link. Every node except the last node in the graph has an adder circuit that adds the maximum bid for the incoming link to the bids on the outgoing link. The last node has a final maximum bid circuit that outputs the optimal value for the combinatorial auction.

A combinatorial auction circuit has now been developed that can output the optimal value for the auction graph, but it also needs to output what bidders won what goods and the price for each bidder. Every maximum bid circuit can output not only the maximum bid for that link but also the maximum bidder. At every link in the graph the maximum bid circuit outputs what bidder won the subset of goods represented by this link and what price they bid. These values are combined for every link in a path by OR′ing together the outputs of every link. The outputs of every path are AND′ed with the output of the final maximum bid circuit and OR′ed together over all the paths. Every winning bidder and price not on the optimal path is reset by AND′ing it with the output of the final maximum bid circuit which outputs 0 for every path except the optimal one.

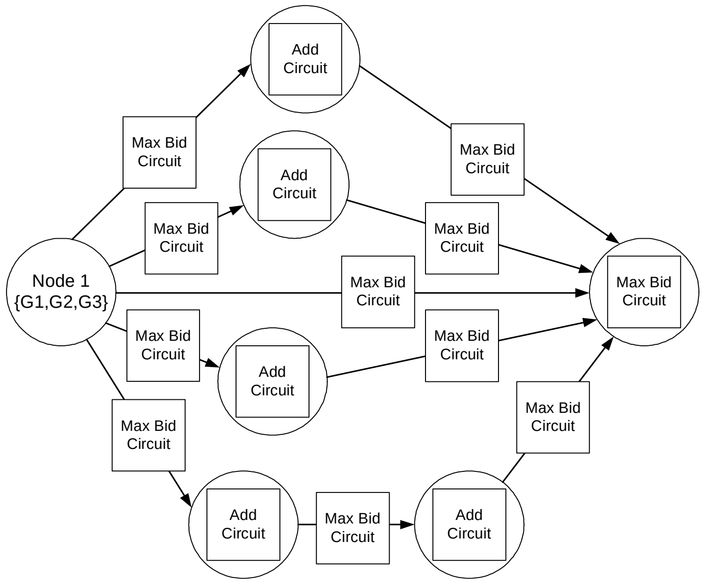


Figure 4.6: Circuits on an Auction Graph

4.3.2 The Combinatorial Auction Circuit Creation Algorithm

Now the design of an auction circuit to compute the outcome of combinatorial auctions was complete, the next step was to write an algorithm that outputs a combinatorial auction circuit based on the number of bidders, the maximum price, and the number of goods. Based on the input parameters, the algorithm first constructs an auction graph that represents the auction. Figure 4.5 shows an example auction graph for 3 goods. The algorithm then constructs the auction circuit by iterating through the auction graph and constructing sub-circuits of maximum bid circuits for every link in the auction graph and add circuits for every node. The inputs and outputs of these sub-circuits are connected in such a way that the circuit computes the result of the combinatorial auction as described in the previous section. This algorithm has been implemented in Java and the circuit created as a Java object. Using this algorithm the garbled circuits auction protocol can be used to conduct combinatorial auctions, something that was not possible based on the original garbled circuits paper.

4.4 Verification

Naor, Pinkas, and Sumner suggest several verification techniques to cover some of the potential problems with this protocol [36]. The problems addressed by the verification process are:

- A malicious auction issuer sending a garbled circuit to the auctioneer that does not correctly compute the result of the auction and could declare an arbitrary bidder as the winner. If the auction issuer sends x different copies of the garbled circuit to the auctioneer who, with the help of the auction issuer, opens and checks x-1 of the circuits to make sure they correctly execute the auction, then an auction issuer who is sending corrupt circuits could be found with probability $(x-1)/x$.

- A malicious auction issuer can change bids when using the original proxy oblivious transfer (POT) protocol of the original garbled circuit auction protocol. This is addresses by using the verifiable proxy oblivious transfer (VPOT) protocol that was suggested by Juels and Szydlo [28].

- A malicious bidder could submit an invalid bid. An invalid bid would be found by the VPOT protocol and could be replaced with a place sitting bid of 0 without restarting the auction.

- A malicious auctioneer could alter the bid values as they come through from the bidders before they forward them on to the auction issuer. If the auction issuer publishes a hash of all the bid values it receives via the auctioneer, bidders can check if the correct hash values for their bids are present.

- The auctioneer could execute another circuit rather than the one sent by the auction issuer. This can be detected if the auction issuer publishes a signed translation table that provides the possible outputs of the circuit put through a one way function. The bidders or auctioneer can then check that the outputs the auctioneer lists put through the same one way function are present in the translation table.

These techniques can be used to provide a group verification process for the garbled circuits auction protocol as long as the auctioneer and auction issuer do not collude.

4.5 Security Analysis

Garbled circuits are privacy preserving as long as the auction issuer and auctioneer do not collude. It is secure against a passive adversary when the adversary is either a bidder, auctioneer, or the auction issuer. The protocol can be extended to detect active adversaries who are not keeping to the correct protocol using the verification techniques in the previous section.

In a secure garbled auction, the auctioneer should not be able to work out the intermediate values of the circuit. The random permutation of wire i, $\pi_i : b_i \rightarrow c_i$, prevents the auctioneer from learning these intermediate values as it will know the c_i value but has no way of knowing what b_i value it maps to.

If the auctioneer knows the garbled inputs and output of a node, it cannot find out the other garbled output of the node. The random function F masks the other values and makes them appear random.

If the number of outputs of a node is greater than 1 a different input to the random function must be used for each output. If an identifier I is assigned to each node then $F_W(c, I)$ can be used for masking instead of $F_W(c)$ [36]. This results in a separate gate table for every output node.

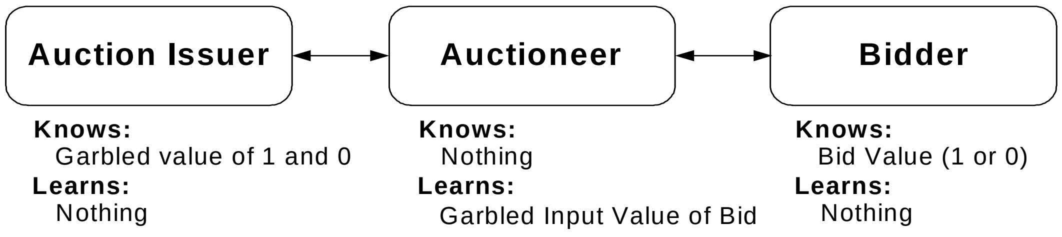


Figure 4.7: VPOT protocol

The auctioneer should also be prevented from learning the actual input values of the bidders.The verifiable proxy oblivious transfer protocol (VPOT) [29] is a multi party protocol involving the bidder, auctioneer, and auction issuer and is illustrated in Figure 4.7. The auction issuer knows the garbled value of the input wires, the bidders knows the actual input, and the auctioneer learns the garbled value of the input only. The VPOT protocol prevents the auctioneer or auction issuer from learning the actual inputs of the bidder, as well as providing verification that the bidder, auctioneer, and auction issuer carried out the protocol correctly.

4.6 Summary

This chapter has introduced garbled circuits and the garbled circuit auction protocol by Naor, Pinkas, and Sumner [36]. The construction of a Boolean circuit to compute the result of combinatorial auctions has then been detailed. Using the combinatorial auction circuit developed in this chapter, the garbled circuit auction protocol is able to compute the result of combinatorial auctions where previously it had only been used to compute the results of first price and (M+1) priced auctions.

Chapter 5

Group Verification Protocol

The garbled circuits auction protocol of the previous chapter uses two party trust. Greater robustness can be achieved by using a threshold trust model where a group of auctioneers conducts the auction protocol and the protocol can still complete if one of the auctioneers fails or is malicious. the homomorphic auction protocol by Suzuki and Yokoo [53] is a threshold trust auction protocol that can conduct combinatorial auctions and keep losing bid values secret but is not verifiable. Verification of the homomorphic auction protocol provides auction participants with greater confidence in the result of the auction. The first iteration of our verification protocol for the homomorphic auction protocol is a group verification protocol. A group verification protocol can only be verified by participants taking part in the protocol. In this case, the auctioneers all provide zero knowledge proofs of the actions they take to execute the auction protocol and they can verify the actions taken by other auctioneers. If an auctioneer detected one of the other auctioneers deviating from the correct behaviour for the auction protocol, they could publish information on the deviating auctioneer publicly.

A malicious auctioneer could claim that a correct auctioneer has behaved incorrectly when it has not. To prevent this, at least a quorum of auctioneers is required to claim that an auctioneer has deviated from the auction protocol. This quorum value is the same value used in the threshold encryption. This means that unless at least a quorum of auctioneers are malicious, losing bid values are kept secret and bidders can have confidence that the auction protocol executed correctly unless indicated otherwise by the auctioneers. In the case of quorum auctioneers indicating that one of the actions taken by one of the auctioneers has failed the verification, the auction protocol can either continue ignoring any values returned by the offending auctioneer, or the auction result can be invalidated and the auction started again.

5.1 Threat Model

The provers and verifiers in our verification protocol are assumed to be polynomially-bounded active adversaries that may try and prove incorrect assertions. For example, an active adversarial prover may try and convince an honest verifier that they have correctly computed an incorrect auction result. It is assumed that any number of polynomially-bounded active adversaries may be colluding together to disrupt the verification protocol as long as there are less than the quorum or threshold value t used in the (t, n) threshold scheme. Different parties may collude, so a bidder and an auctioneer may collude together.

It is assumed that any party can get a copy of any public message sent between the auction participants and use it to try and break the verification protocol. The verification protocol does not address the case where a malicious party controls the communication channel and prevents messages from reaching a particular participant effectively performing a denial of service attack on that party, alters the messages in transit, or replays messages. Altered messages will likely fail verification and could be used to reduce the confidence in an honest prover, however schemes such as digital signatures of messages can be used to prevent this kind of attack but are outside the scope of this thesis.

5.2 Security Goals

The security goals for the verification protocol are now reviewed in the context of a group verification protocol for the homomorphic auction protocol.

1. Auctioneers should be able to verify the actions of the other participants in the auction protocol giving a high confidence that they have correctly executed the steps in the auction protocol.

2. Verification of the auction protocol should reveal no information other than what is revealed by the auction protocol.

3. It should be computationally infeasible for a bidder to submit an invalid bid that passes the verification checks.

4. It should be computationally infeasible for an auctioneer to not count all the bids and pass the verification checks. For the homomorphic auction protocol, it should be computationally infeasible for an auctioneer to not count a bid when calculating the maximum bid for a node and to pass the verification checks.

5. It should be computationally infeasible for an auctioneer to announce an incorrect winning bidder(s) or price(s) and pass the verification checks. For the homomorphic auction protocol, it should be computationally infeasible for an auctioneer to do an incorrect shift and randomise, or incorrectly compute the optimal path once the optimal value is found and to pass the verification checks.

5.3 Threshold El-Gamal Decryption

Threshold El-Gamal encryption is used in the homomorphic auction protocol so that a single auctioneer cannot decrypt all the losing bids for a link in the auction graph. The El-Gamal secret key is shared among a group of n members $x \rightarrow (x_1, ..., x_n)$ using Shamirs secret sharing scheme [47]. Encryption still involves using a public key $y = g^x$ and a random value r to calculate $(A, B) = (g^r, y^r M)$. To perform a threshold decryption of a value when using a (t, n) threshold encryption scheme requires a group I of participants with at least t members. Every participant in I knows the indexes i of the other members in the group. In a $(3, 5)$ sharing scheme the group I could be made of parties with indexes 2, 3, and 5. Every member of I then follows the following steps to calculate A^x used in the decryption where $m = B/A^x$:

- Every member i calculates and publishes among the group $z_i = A^{x_i}$.

- All members compute on their own the λ-coefficients of the Lagrange interpolation of the indexes i in the group I.

- Every member can then compute on their own

$$A^x = \prod_{i \in I} z_i^{\lambda_i} = \prod_{i \in I} (A^{x_i})^{\lambda_i} = A^{\sum_{i \in I} \lambda_i x_i} = A^x$$

  using Lagrange interpolation to calculate $x = \sum_{i \in I} \lambda_i x_i$.

- The decryption can then be completed by all the members on their own calculating $M = B/A^x$ as in standard El-Gamal decryption.

When every member of the group I correctly follows the protocol the decryption will succeed and correctly decrypt the cipher-text.

5.3.1 Verifiable Threshold El-Gamal Decryption

When using threshold El-Gamal decryption, if one of the members of the group publishes an incorrect z_i value the correct plain-text will not be decrypted. If the individual public keys of the secret keys $(y_1 = g^{x_1}, y_2 = g^{x_2}, ..., y_n = g^{x_n})$ can be published with the group public key y, the members in the group doing the decryption I can publish zero knowledge proofs that they correctly computed the z_i values. Every member i will publish a zero knowledge proof of equality of logarithms from section 3.2.2 by publishing a proof that they know the value x_i such that $y_i = g^{x_i}$ and $z_i = A^{x_i}$.

By publishing a zero knowledge proof that the z_i value is correctly computed, a malicious participant publishing incorrect z_i values will be detected with a high probability and can be removed from the group. As long as less than t participants of the secret sharing scheme are malicious, the decryption can be carried out correctly even in the presence of malicious participants.

5.3.2 Completeness

Each participant in the group doing the decryption has to publish a proof of equality of discrete logarithms. As the proof of equality of logarithms is shown to be complete in Section A.2.1, the verifiable threshold El-Gamal decryption must also be complete. To put it another way, suppose the verifiable threshold decryption was not complete, then one of the proof of equality of discrete logarithms must not be complete, but these are known to be complete.

5.3.3 Soundness

Suppose a cheating prover is trying to publish a proof that they know a value x_i where $y_i \neq g^{x_i}$ or $z_i \neq A^{x_i}$. This is analogous to the situation of a cheating prover in the proof of equality of logarithms as shown in Section A.2.2 so the soundness is $1/q$.

5.3.4 Zero Knowledge

To check the zero knowledge property of the proofs of equality used in threshold decryption, a simulator S can be constructed that completes the following steps on common input p, q, g, the encrypted item (A, B), the set of participants I, the shares $z_1, ..., z_n$, and the public keys $y_1, ..., y_n$ in the random oracle model:

- For every item i in the set of participants I, S does the following:
  - S outputs the transcript of the simulator for the equality of logarithms shown in Section A.2.3 using as input p, q, g, A, y_i, and z_i which has output a_i, b_i, r_i.

The verifier then checks that for every item in the set of participants I, $g^{r_i} = a_i y_i^{c_i}$ and $A^r = b_i z_i^{c_i}$.

5.4 Zero Knowledge Proof of a Valid Bid Vector

A valid bid vector should be of the form:

$$bid_{valid} = E(Z), E(Z), E(Z), E(1), E(1), E(1)$$

All the items in the bid vector should be encryptions of 1 or the publicly known value Z. A bid vector should also contain no gaps where an encrypted 1 is between two encrypted Z values. An invalid bid vector with a gap would be of the form:

$$bid_{gap} = E(Z), E(Z), E(1), E(Z), E(1), E(1)$$

To assist in the zero knowledge proof of a valid bid vector an alternate form of the bid vector is used. Suppose every item i in the alternate bid vector alt for a value v is calculated with the formula:

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

So a bid vector of length 6 and value 3 would be

$$E(1), E(1), E(Z), E(1), E(1), E(1).$$

To prove in zero knowledge that a bid vector in the alternate form is valid, we publish proofs that every item in the bid vector is a 1 or a Z. The product of all the items in the bid vector is then computed and is proved to decrypt to Z. The bid vector is then converted to standard form using a technique called integrating a bid vector [1].

When the bidders publish their alternate form bid vectors, they also publish a proof an encrypted item decrypts to 1 or Z from section 3.2.3 for every item in the vector. The auctioneers can then verify that every item in the alternate form bid vector is an encryption of 1 or Z.

The auctioneers each independently compute the product of all the items in the bid vector. This product can be computed by any party due to the homomorphic nature of El-Gamal encryption. The auctioneers publish their shares of the decryption of the product as well as zero knowledge proofs that their shares are correctly computed as shown in Section 5.3.1. The auctioneers can then verify that the product of the items was correctly computed and, using Lagrange interpolation, that the product of the items decrypts to Z.

Now that zero knowledge proofs that the alternate form of the bid vector is valid have been constructed, the alternate bid vector is integrated to convert it to the standard form of the bid vector used in the homomorphic auction protocol. To integrate a bid vector one of the auctioneers computes the following function on the items of the alternate bid vector alt of length l starting with the item l and going down to item 1 to get the standard bid vector bid:

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

This formula converts the alternate form of the bid vector to the standard form of the bid vector and is verifiable due to the homomorphic nature of El-Gamal encryption.

5.4.1 Completeness

There are four steps in this zero knowledge proof that need to be shown to be complete:

- The proofs published by the bidders that every item in the alternate bid vector is an encryption of a 1 or a Z were shown to be complete in Section 3.2.3.
- Computing the product of the items is complete due to the homomorphic nature of El-Gamal encryption. It is simply a multiplication of items in the bid vector.

- When computing and publishing shares of a threshold decryption the published proofs are proofs of the equality of discrete logarithms which were shown to be complete in Section 3.2.2.
- Converting the alternate vector into standard form is complete due to the homomorphic nature of El-Gamal encryption.

5.4.2 Soundness

The soundness of the zero knowledge proofs used is examined in four steps:

- The proofs published by the bidders that every item in the alternate bid vector is an encryption of a 1 or a Z are shown in Section 3.2.3 to have a soundness of $1/q$.
- Computing the product of the items in the alternate form of the bid vector is completely sound as any party can repeat the multiplication of the items and confirm this has been done correctly. Unless the verifier has an error in their computation, they will always catch a cheating auctioneer in this step.
- When computing and publishing shares of a threshold decryption the proofs published are proofs of equality of discrete logarithms which were shown in Section 3.2.2 to have a soundness of $1/q$.
- Converting the alternate vector into standard form is completely sound as any party can repeat the multiplication of the items and confirm this has been done correctly.

If an invalid bid vector has been proven to be valid then one of two things must have happened. Either the bidder publishing the vector has managed to publish a proof that an item is an encryption of a 1 or a Z when it is not, or the decryption of the shares published by the auctioneers is shown to decrypt to Z when it does not.

In the first instance a bidder could prove an item decrypts to 1 or Z when it does not with probability $1/q$. Even if the bidder managed to achieve this for one item in the bid vector, when the auctioneers decrypt the product of the items it will no longer equal Z so the bidder would have to change two items in the bid vector and set them so that the product of the items in the bid vector is still Z. The chance of a bidder being able to prove that two items in the bid vector decrypt to 1 or Z when they do not is $1/2q$.

In the second instance, the chance of an auctioneer in the group being able to publish an incorrect share that would cause a problem in the decryption while still proving it is correct is again $1/q$. If more than one auctioneer was malicious, all the malicious auctioneers would have to publish proofs that their shares are correct when they are not, again with a chance of $1/q$ for each share. This is true even if more than the threshold of auctioneers is malicious.

5.4.3 Zero Knowledge

To check the zero knowledge property of this proof, a simulator S can be constructed that completes the following steps on common input p, q, g, y, $AlternateBid = ((A_1, B_1), ..., (A_n, B_n))$, the share values $z_1, ..., z_n$, the public keys $y_1, ..., y_n$, the set of participants I, and Z in the random oracle model:

- S completes the steps of the simulator in Section 3.2.3 for every item (A_i, B_i) in the alternate bid vector $AlternateBid$ with inputs p, q, g, y, A_i, B_i, and Z.
- S computes the product $Prod$ of all the items in $AlternateBid$.
- S completes the steps of the simulator in Section 5.3.4 with input p, q, g, (A_i, B_i), I, $z_1, ..., z_n$, and $y_1, ..., y_n$.
- S computes the standard bid vector Bid from $AlternateBid$.
- S outputs the proof transcripts from the previous steps, $Prod$, and Bid.

5.5 Zero Knowledge Proof of the Maximum Bid

To calculate the maximum bid for a set of goods, the product of all the bid vectors is taken. This product vector is then decrypted from right to left to find the first value in the product vector that does not decrypt to 1. This reveals the maximum bid while keeping private the values of the lower bids. To prove in zero knowledge to the group of auctioneers conducting the auction that this is done correctly, all members of the group will need to compute the product vector on their own. They can then decrypt the items from right to left and publish zero knowledge proofs that this decryption has been done correctly using verifiable threshold El-Gamal decryption in Section 5.3.1. Every member of the group can then each independently verify the shares they have received from the other auctioneers and decrypt the values in the bid vector with a high certainty that this decryption is the correct plain-text.

As this proof is composed of only verifiable decryptions of items, the completeness, soundness, and zero knowledge properties are the same as for verifiable decryption.

5.6 Zero Knowledge Proof of Shift and Randomise

Once the maximum bid m on a link in the path has been calculated, the bid vectors in the next link are shifted right by m places which are taken by encrypted Z values. The shifted bid vector is then randomised and published. The auctioneer that calculates the shifted and randomised bid vector of length l can prove this has been done correctly in zero knowledge by conducting the shift and randomise using the following steps:

- The auctioneer calculates m encryptions of Z to shift the bid vector and publishes zero knowledge proofs that these m items are encryptions of Z. The auctioneer publishes zero knowledge proofs of equality of logarithms by publishing a proof that they know the value r such that $A = g^r$ and $B/Z = y^r$.

- The auctioneer then shifts the new bid vector right by m places and adds the m encryptions of Z that they have proved in zero knowledge decrypt to Z and publishes this shifted bid vector to the other auctioneers in the group.

- Auctioneer calculates l encryptions of 1 and publishes zero knowledge proofs that these l items are encryptions of 1. The auctioneer publishes l zero knowledge proofs of equality of logarithms by publishing proofs that for every item l they know the value r such that $A = g^r$ and $B = y^r$.

- The auctioneer then publishes these l encryptions of 1 to the other auctioneers.

- The auctioneer can then calculate and publish the shifted and randomised bid vector by multiplying the items in the shifted vector by the l encryptions of 1.

As this proof is composed of proof of equality of discrete logarithms the completeness, soundness, and zero knowledge properties are the same as for the zero knowledge proof of equality of discrete logarithms shown in Appendix A.2.

5.7 Example Verifiable Combinatorial Auction

Figure 5.1 illustrates a simple threshold combinatorial auction for two goods with two bidders. There is a group of auctioneers responsible for calculating the auction result at each node other than the start node of the graph. In this example, a $(2, 3)$ threshold scheme is being used. The correct winner of the auction is bidder one with a price of 5 for goods one and two together.

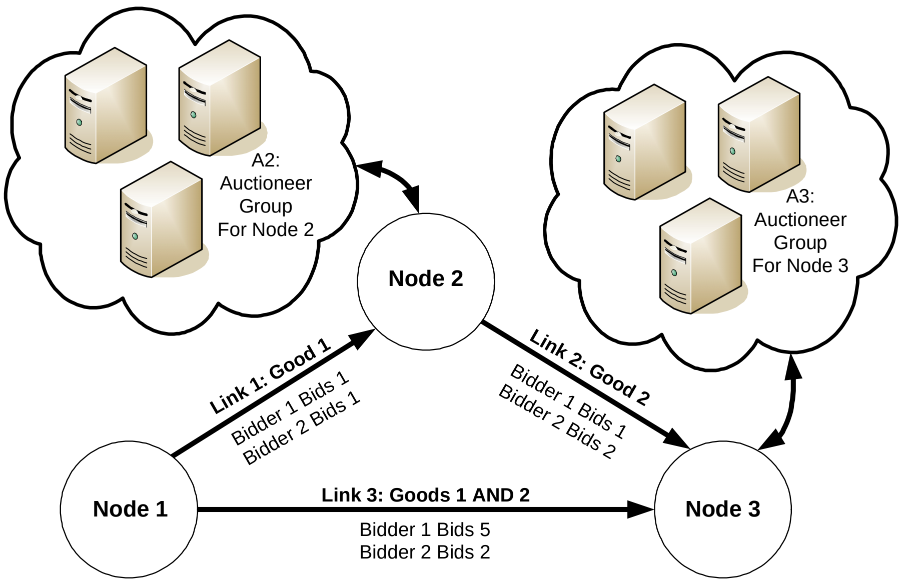


Figure 5.1: Simple Threshold Combinatorial Auction

To perform a group verification that the auction has correctly taken place the following actions are executed in conjunction with the actions executed to calculate the winner of the auction. If a problem is reported to a public bulletin board by at least t auctioneers from the (t, n) threshold scheme, then the auction has failed verification.

1. Bids are encrypted and published in alternate form for every link using the public key of the link. Bidders also publish zero knowledge proofs that they are valid. Auctioneers for the link check alternate form bid vectors before

converting them to the standard bid vector form. The process for publishing and verifying the bid for bidder one on link three is shown in figure 5.2 and is divided in to five steps:

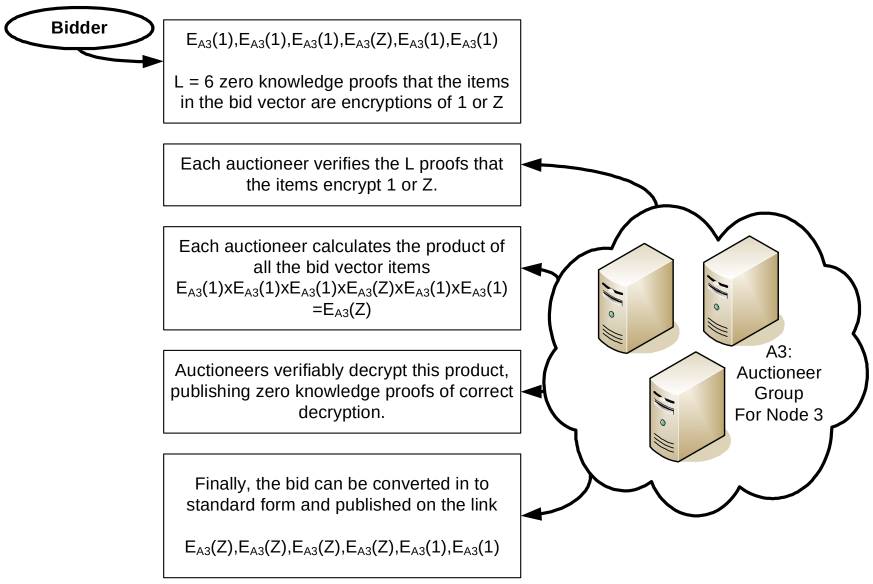

Bidder
$E_{A3}(1),E_{A3}(1),E_{A3}(1),E_{A3}(Z),E_{A3}(1),E_{A3}(1)$
L = 6 zero knowledge proofs that the items in the bid vector are encryptions of 1 or Z
Each auctioneer verifies the L proofs that the items encrypt 1 or Z.
Each auctioneer calculates the product of all the bid vector items $E_{A3}(1)xE_{A3}(1)xE_{A3}(1)xE_{A3}(Z)xE_{A3}(1)xE_{A3}(1)$ $=E_{A3}(Z)$
Auctioneers verifiably decrypt this product, publishing zero knowledge proofs of correct decryption.
Finally, the bid can be converted in to standard form and published on the link
$E_{A3}(Z),E_{A3}(Z),E_{A3}(Z),E_{A3}(Z),E_{A3}(1),E_{A3}(1)$
A3: Auctioneer Group For Node 3

Figure 5.2: Bid Verification

- The bidder encrypts the bid vector for the value it wants to bid for this link or combination of goods with the public key of the group of auctioneers responsible for the link. In our example, group $A2$ is responsible for link two, and group $A3$ is responsible for links one and three. When encrypting every item in the alternate bid vector the bidder also constructs a zero knowledge proof that the item it is encrypting decrypts to a 1 or a Z. This encrypted alternate bid vector as well as the zero knowledge proofs are sent to the auctioneer group.
- The auctioneers in the group verify the zero knowledge proofs that each item is a 1 or a Z sent by the bidder with the bid.
- Each auctioneer computes the product of all the items in the alternate bid vector. This can be done correctly by any auctioneer due to the homomorphic nature of the encryption.
- The auctioneers then each do a verifiable decryption of the product of the items of the bid vector. If the decrypted product does not decrypt to Z then this is not a valid bid vector.

- If this is a valid bid vector, it can be converted in to standard form by an auctioneer and published on the link. Any other auctioneer can check this has been done correctly due to the homomorphic nature of the encryption.

2. Once encrypted bids have been submitted and verified, the auction graph appears as in Figure 5.3.

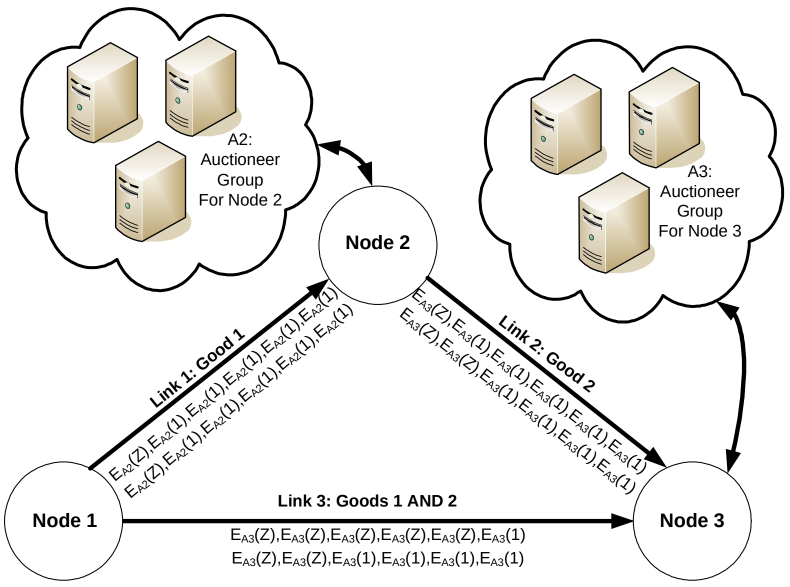

A2: Auctioneer Group For Node 2
A3: Auctioneer Group For Node 3
Node 2
Node 1
Node 3
Link 1: Good 1
$E_{A2}(Z),E_{A2}(1),E_{A2}(1),E_{A2}(1),E_{A2}(1),E_{A2}(1)$
$E_{A2}(Z),E_{A2}(Z),E_{A2}(1),E_{A2}(1),E_{A2}(1),E_{A2}(1)$
Link 2: Good 2
$E_{A3}(Z),E_{A3}(1),E_{A3}(1),E_{A3}(1),E_{A3}(1),E_{A3}(1)$
$E_{A3}(Z),E_{A3}(Z),E_{A3}(1),E_{A3}(1),E_{A3}(1),E_{A3}(1)$
Link 3: Goods 1 AND 2
$E_{A3}(Z),E_{A3}(Z),E_{A3}(Z),E_{A3}(Z),E_{A3}(Z),E_{A3}(1)$
$E_{A3}(Z),E_{A3}(Z),E_{A3}(1),E_{A3}(1),E_{A3}(1),E_{A3}(1)$

Figure 5.3: Simple Threshold Combinatorial Auction with Encrypted Bids

3. The auctioneers in group $A2$ find the optimal value for their link and then shift and randomise the bids on link 2 by this amount. Zero knowledge proofs are published to the group to prove this has been done correctly. The actions taken by the auctioneers are shown in Figure 5.4 and can be divided in to three steps:

   - The auctioneers first compute the product of all the bid vectors. This can be computed by any party due to the homomorphic nature of the encryption by multiplying the individual items in the bid vector. The first items in both bid vectors are multiplied together and so on until the last items.
   - From right to left, the items in the product bid vector are decrypted to find the first item that does not decrypt to 1. Auctioneers can check

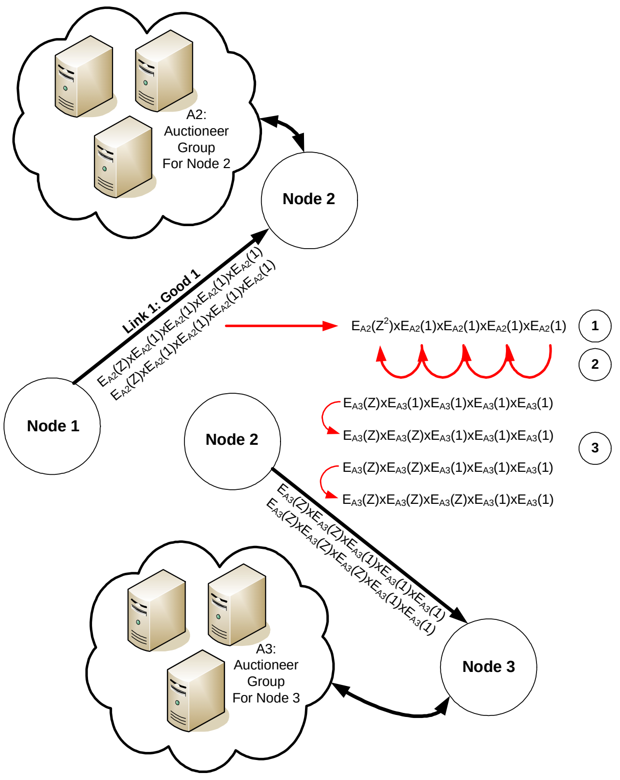

A2: Auctioneer Group For Node 2
Node 2
Link 1: Good 1
E_A2(Z)xE_A2(1)xE_A2(1)xE_A2(1)xE_A2(1)
E_A2(Z)xE_A2(1)xE_A2(1)xE_A2(1)xE_A2(1)
Node 1
E_A2(Z^2)xE_A2(1)xE_A2(1)xE_A2(1)xE_A2(1)
1
2
E_A3(Z)xE_A3(1)xE_A3(1)xE_A3(1)xE_A3(1)
E_A3(Z)xE_A3(Z)xE_A3(1)xE_A3(1)xE_A3(1)
3
E_A3(Z)xE_A3(Z)xE_A3(1)xE_A3(1)xE_A3(1)
E_A3(Z)xE_A3(Z)xE_A3(Z)xE_A3(1)xE_A3(1)
Node 2
E_A3(Z)xE_A3(Z)xE_A3(1)xE_A3(1)xE_A3(1)
E_A3(Z)xE_A3(Z)xE_A3(Z)xE_A3(1)xE_A3(1)
A3: Auctioneer Group For Node 3
Node 3

Figure 5.4: Shift and Randomise by Group $A2$

the decryption is done correctly by doing a verifiable decryption of the items.

- One of the auctioneers then shifts and randomises the bid vector for the next link. In this example, for each of the bid vectors for link 2, the auctioneer will encrypt a Z value with the public key for group $A3$ and publish a zero knowledge proof to the other auctioneers that the item is an encryption of Z. The auctioneer will then shift the bid vectors for link 2 right by one item and add the new encryption of Z to the left hand side of the vector. Finally, the auctioneer will randomise the shifted bid vector by creating $l = 5$ encryptions of 1 using the public key of group $A3$, publish zero knowledge proofs to the other auctioneers that these items are an encryption of 1, and multiply the shifted bid vectors by the encrypted 1 values to randomise them and hide the number of items in the bid vector that have been shifted.

4. The auctioneers in group $A3$ find the optimal value for the auction. The

product of all the bid vectors for link 2 and link 3 are multiplied together to generate a product vector. The auctioneers for the group $A3$ will then decrypt the items in the product vector from right to left to find the first item that does not decrypt to 1. This value will be the optimal value of this auction, in this auction the optimal value is 4. The decryption is done using verifiable decryption so the other auctioneers of the group can check the share of the decryption published by the other auctioneers.

5. The auctioneers in group $A3$ find the optimal bid on the optimal path. To find the optimal bid, the auctioneers will decrypt the individual bid vectors at the index of the optimal value one by one to find the first bid vector that decrypts to Z. In this example, the auctioneers would decrypt the shift and randomised bids for link 2 as well as the bid vectors for link 3 at the index of the optimal value 4. Only the bid vector from bidder one on link 3 would decrypt to a Z at index 4 showing that the optimal bid is bidder one for goods 1 and 2. When the decryption is being done, zero knowledge proofs of correct decryption can be published to the group $A3$ to provide confidence that the decryption has been done correctly and a correct optimal bid found.

6. The result of the auction is publicly published and the winners notified.

5.8 Summary

This chapter has shown the group verification protocol for the homomorphic auction protocol. The threat model and security goals for the group verification protocol were detailed with the protocol needing to be complete, sound, and zero knowledge. Details were presented on verifiable threshold El-Gamal decryption, the individual actions taken in the homomorphic auction protocol, and a simple example was given. Using the group verification protocol, bidders can have confidence in the auction result as long as less than the threshold t of auctioneers for each group are malicious. The group verification protocol also increases the robustness of the protocol as the auctioneers can find invalid bid vectors before the protocol starts and remove them as well as being able to verify that the decryption shares published by other auctioneers have been performed correctly to prevent the subvertion of the decryption process.

Chapter 6

Public Verification Protocol

The second iteration of our design for a verification protocol for the homomorphic auction protocol is a public verification protocol. Public verification allows any third party to verify the auction even if they did not participate in the auction process. By allowing any party to verify the auction and not just the auctioneers taking part, the verification process is not affected by the number of malicious auctioneers. Even if more than the threshold value t auctioneers are malicious there is still a low chance they could incorrectly execute the auction protocol and publish zero knowledge proofs that are accepted. This is in contrast to the previous group verification protocol which depended on less than the threshold value t of auctioneers being malicious for the verification protocol to work.

The threat model for our verification protocol is restated before presenting the zero knowledge proofs that the bids are valid followed by zero knowledge proofs of the maximum bid and the shift and randomise action. Finally, these proofs are put together and an example is presented of a public verification of a simple two good combinatorial auction.

6.1 Threat Model

It is assumed that the provers and verifiers in our verification protocol are polynomially bounded active adversaries that may try and prove incorrect assertions. For example, an active adversarial prover may try and convince an honest verifier that they have correctly computed an incorrect auction result. It is assumed that any number of polynomially-bounded active adversaries may be colluding together to disrupt the verification protocol regardless of the threshold value. This is in contrast to the group verification protocol where it is assumed that less than the threshold value t of auctioneers are malicious. Different parties may collude, so a bidder and an auctioneer may collude together. It is assumed that any party can get a copy of any public message sent between the auction participants and use it to try and break the verification protocol similarly to the group verification protocol in Section 5.1.

6.2 Security Goals

The security goals for the verification protocol are now reviewed in the context of a public verification protocol for the homomorphic auction protocol.

1. Any party regardless of whether they took part in the auction should be able to verify the actions of the participants in the auction protocol giving a high confidence that they have correctly executed the steps in the auction protocol.

2. Verification of the auction protocol should reveal no extra information other than what is already revealed by the auction protocol.

3. It should be computationally infeasible for a bidder to submit an invalid bid that passes the verification checks.

4. It should be computationally infeasible for an auctioneer to not count all the bids and pass the verification checks. For the homomorphic auction protocol, it should be computationally infeasible for an auctioneer to not count a bid when calculating the maximum bid for a node and pass verification.

5. It should be computationally infeasible for an auctioneer to announce an incorrect winning bidder(s) or price(s) and pass the verification checks. For the homomorphic auction protocol, it should be computationally infeasible for an auctioneer to do an incorrect shift and randomise, or incorrectly compute the optimal path once the optimal value is found and pass verification.

6.3 Zero Knowledge Proof of a Valid Bid Vector

The same zero knowledge proofs and techniques that have been used to prove a bid is valid in the group verification protocol in section 5.4 can be used in the public verification protocol.

Bids are publicly published in the alternate version along with zero knowledge proofs that every item in the bid vector decrypts to a 1 or a Z. The product of all the items in the alternate bid vector is calculated and the auctioneers publicly publish their shares of the decryption of this product along with zero knowledge proofs that their share is correct. Any party can then check the decryption equals Z using LaGrange interpolation. The alternate bid vector can then be converted in to a standard bid vector, an action that any party can verify due to the homomorphic nature of El-Gamal, and publicly published.

6.4 Zero Knowledge Proof of the Maximum Bid

To calculate the maximum bid for a node, the auctioneers compute the item wise product of all the bid vectors on the incoming edges. This can be done due to the homomorphic nature of the encryption. The auctioneers then decrypt the items in the product vector from right to left until the first item that does not decrypt to 1 using the verifiable decryption from Section 5.3.1 where the zero knowledge proofs that the decryption is done correctly are only published to other auctioneers in the group. The first item that does not decrypt to 1 is the maximum bid value m for the node. Once the maximum value m has been found, the auctioneers decrypt all the individual bid vectors at index m to find the maximum bid vector Bid_{max} using the verifiable decryption from Section 5.3.1 where the zero knowledge proofs that the decryption is done correctly are again only published to other auctioneers in the group.

To publicly prove in zero knowledge that a bid vector Bid_{max} is the maximum bid from the set of bid vectors, one of the auctioneers from the group uses a publicly verifiable shuffle of encrypted values from section 3.2.5. The bid vectors are all shuffled using the same permutation that is known to the auctioneer that does the shuffle but remains unknown to everyone else. This permutation is chosen so that the first item in the shuffled vector is the mth item from the original bid vector. The auctioneers for the group then publicly publish their shares of the decryption of the first items in the shuffled bid vectors along with zero knowledge proofs that these shares are correct from Section 5.3.1. Any party can then compute the Lagrange interpolation on the shares to decrypt the items. The first item in the maximum bid vector Bid_{max} should decrypt to Z while the first items for all the other bid vectors decrypt to 1. As all the other bids are shown to decrypt to 1 and it can be verified that the shuffle of the encrypted values was done correctly this verifies that all the bids have been counted in the auction and that the winning bid is the maximum bid. Because of the shuffle applied to the bid vectors, this verification process reveals no information about the bids other than what can already be deduced from the assertion that Bid_{max} is the maximum bid. The shuffle applied to the bid vectors hides the value of the maximum bid.

If there is a tie for the maximum bid and the shuffled first item of more than one bid decrypts to Z, it can be proved that it is one of the set of maximum bids by performing the following steps for every bid vector $Other_{max}$ with a shuffled first item that decrypts to Z:

- Calculate the product of all the items in $Other_{max}$. For example, if the vector is $E(Z), E(Z), E(1)$ the product will be $E(Z^2)$.
- The auctioneers for the group now divide the product of all the items in Bid_{max} by the product computed in the previous step.
- The auctioneers then publicly publish their shares of the decryption of the result from the division in the previous step together with zero knowledge proofs that these shares are correct from Section 5.3.1. Any party can then compute the Lagrange interpolation on the shares to decrypt the items.
- If the decryption of the division is 1 then the vector $Other_{max}$ is equal to the vector Bid_{max}.

These steps will only need to be performed in the case that there is a tie-break and more than one maximum bid.

6.4.1 An Example

Suppose there are three bids:

- Bid 1: $E(Z), E(Z), E(Z), E(1)$.
- Bid 2: $E(Z), E(Z), E(1), E(1)$.
- Bid 3: $E(1), E(1), E(1), E(1)$.

Bid 1 is the maximum bid. To prove this one of auctioneers applies a permutation to the bids say $\pi = (3, 4, 1, 2)$. After this permutation, the three bids will be as follows:

- Bid 1: $E(Z), E(1), E(Z), E(Z)$.
- Bid 2: $E(1), E(1), E(Z), E(Z)$.
- Bid 3: $E(1), E(1), E(1), E(1)$.

The auctioneer that performed the shuffle then publishes a zero knowledge proof that the permutation was correctly applied and that the same permutation was applied to all the bid vectors. The auctioneers then publish shares of the decryption of the first items in the shuffled vectors along with proofs that these shares are correct. Using the shares, a verifier can check that the first item in the shuffled Bid 1 vector decrypts to Z, while the first item in the other bid vectors decrypt to 1. As the vectors have been shuffled this reveals no other information other than the maximum bid to any verifiers.

By using the zero knowledge proof of the maximum bid for a node, both the auctioneer and any verifying party will learn the index of the bidder that bid the maximum amount on this link. This is extra information that, unless this link is on the optimal path, would normally remain unknown. To prevent this information leakage, the auction protocol can be run in an anonymous fashion where bidders use pseudo names to submit bids and the true identity of the bidders remain secret unless they try and repudiate on their bid. Various techniques for providing bidder anonymity were briefly described in Section 2.6.

6.4.2 Completeness

The completeness of this proof can be divided in to two sections. The completeness of the zero knowledge proof of the shuffle and the completeness of the verifiable threshold decryption applied to the first items in the shuffled bid vectors. The completeness of the shuffle is shown in Section 3.2.5 and the completeness of the threshold decryption of items is shown in Section 5.3.1.

6.4.3 Soundness

Again, the soundness of this proof can be divided in to the soundness of the shuffle and the soundness of the decryption. From Sections 3.2.5 and 5.3.1 the soundness of this proof is shown to be $1/q$.

6.4.4 Zero Knowledge

To check the zero knowledge property of this proof, a simulator S can be constructed that completes the following steps on common input p, q, g, y, the bid vectors from the incoming links, the shuffled bid vectors, the share values $z_1, ..., z_n$ for each decryption of the first items in the shuffled bid vectors, the public keys $y_1, ..., y_n$, and the set of participants I used in the decryption in the random oracle model:

- S outputs the transcript of the simulator of the verifiable shuffle from [17] with the modifications from Section A.5.3.
- S outputs the transcript for the verifiable threshold El-Gamal shown in Section 5.3.4 on input p, q, g, the first item in the shuffled bid vector (A, B), I, $z_1, ..., z_n$, and $y_1, ..., y_n$ for every shuffled bid vector.

This simulator does not work in the case that Bid_{max} is one of a set of maximum bids. A simulator for this situation would need some extra steps where the simulator would calculate the product of the other maximum bid vectors $Other_{max}$ and then would divide these by the product of Bid_{max}. The simulator would then need to output a transcript that the shares published to do this decryption were correct again using the steps shown in Section 5.3.4.

6.5 Zero Knowledge Proof of Shift and Randomise

Once the auctioneers for a node have published a zero knowledge proof of the maximum bid Max with value m, they will have to shift and randomise the bids on the outgoing links. To shift and randomise a bid vector Old of length l, one of the auctioneers will need to make m encryptions of Z, shift the bid vector Old right by m places inserting the new encryptions of Z and randomise the vector by multiplying every item in the bid vector by different encryptions of the value 1 to create New.

The public zero knowledge proof that this has been done correctly is divided in to two parts. The first part of the proof involves the auctioneers proving that the new shifted bid vector New is a valid bid vector. In the second part of the proof, the auctioneers publish proofs that New has the same value as Old shifted right by the amount of Max.

Before proving the new shifted bid vector New is valid, it is converted from the standard form to the alternate form presented in Section 5.4. The alternate form bid vector Alt is formed item wise by computing from right to left:

$$Alt_i = \begin{cases} New_i & \text{if } i = l \\ New_i / New_{i+1} & \text{otherwise} \end{cases}$$

So if, for example, $New = E(Z), E(Z), E(Z), E(1), E(1), E(1)$ then the conversion will produce $Alt = E(1), E(1), E(Z), E(1), E(1), E(1)$ which is the bid vector New in the alternate format. Anyone can verify this step has been done correctly as both the bid vector New and the alternate form Alt are publicly published and the computation can be checked due to the homomorphic nature of the encryption.

To publish a zero knowledge proof that the bid vector Alt is valid one of the auctioneers conducts a random verifiable shuffle of the items in Alt using the 'publicly Verifiable Shuffle of Encrypted Values' from section 3.2.5. After shuffling the values in Alt, the auctioneers publish their shares of the decryption of every item in the shuffled alternate form bid vector along with zero knowledge proofs that the shares are valid using the techniques from Section 5.3.1. A verifier can then verify the shares and use Lagrange interpolation to decrypt the items in the shuffled alternate bid vector. Every item in the shuffled alternate bid vector should decrypt to 1 except for the one Z. This decryption of the items in the shuffled vectors reveals no information about the values of bids as the shuffle hides the position of the Z value in the alternate bid vector.

Finally, one of the auctioneers publishes the product of all the items in the vector New as $Product_{New}$. They also publish the product of the items in the vector Old multiplied by the product of all the items in the vector Max as $Product_{OldMax}$. Any party can verify that these products have been correctly computed due to the homomorphic nature of the encryption. One of the auctioneers then computes $result = Product_{New}/Product_{OldMax}$. The auctioneers then publish their shares of the decryption of $result$ along with zero knowledge proofs that these shares have been correctly computed. any party can then verify the shares published by the auctioneers and that $result$ decrypts to 1 using Lagrange interpolation. If $result$ decrypts to 1, then the number of encrypted Z values in the vector New is equal to the number of encrypted Z values in the vectors Old and Max. If all the bid vectors are valid this proves that the resulting vector New is the result of a shift and randomise on Old by the value of Max.

6.5.1 Completeness

The completeness of this proof is examined in two sections. The first section addresses the completeness of the proof that New is a valid bid vector, then the completeness of the proof that New is equal to Old shifted right by the value of Max is addressed.

The first step of the proof that New is a valid bid vector is complete as it is multiplication of encrypted values. The completeness of the verifiable shuffle is shown in Section 3.2.5. The decryption of all the shuffled items in the shuffled vector is also complete as shown in Section 5.3.1.

The completeness of the calculation of the products of New and Max and Old are complete as it is multiplication of encrypted values. The calculation of

$$result = Product_{New}/Product_{OldMax}$$

is also complete as it is again multiplication of encrypted values. The completeness of proving that $result$ decrypts to 1 follows from the completeness of the verifiable decryption shown in Section 5.3.1.

6.5.2 Soundness

The soundness of the proof is examined in two steps starting with the proof that the bid vector New of length l is a valid bid vector and then looking at the proof that $result$ decrypts to 1.

Suppose a cheating prover wants to construct a bid vector that will be accepted as a valid bid vector when it is not valid. Given that every item in the alternate bid vector is decrypted to check if it is a 1 or a Z where only one Z can be present for it to be valid, one approach for the cheating prover is to try and construct New in such a way that after it is converted in to alternate form every item still decrypts to 1 except for one Z while not being a valid bid vector. As the item at index l in the alternate vector is just the item at index l from the vector New, then the lth item of New, New_l, must be $E(Z)$ or $E(1)$. The item at index $l-1$ in the alternate bid vector is $Alt_{l-1} = New_{l-1}/New_l$. If $New_l = E(Z)$ then Alt_{l-1} must be equal to $E(1)$ as only one Z is allowed in the alternate bid vector, but if $Alt_{l-1} = E(1)$ and $New_l = E(Z)$ then $New_{l-1} = New_l * Alt_{l-1} = E(Z) * E(1) = E(Z)$ so New_{l-1} must be $E(Z)$ for all subsequent items in the bid vector. If $New_l = E(1)$ then Alt_{l-1} can be equal to $E(Z)$ or $E(1)$ which means that New_{l-1} can be equal to $E(Z)$ or $E(1)$. This argument can be continued for all items in the bid vector New to show that New must be a valid bid vector if the decryption is done correctly.

If a cheating prover wants to do an incorrect decryption of a shuffled item to show it decrypts to a 1 or a Z when it does not, or to change the items when doing the shuffle so that the items in Alt and the shuffled Alt are different then they have to cheat the proof of equality of logarithms or the verifiable shuffle of encrypted items. The chance of doing this is $1/q$ as shown in Sections 3.2.5 and 5.3.1 respectively.

The calculation of the products of New and Max and Old are sound as they are multiplication of encrypted values. The calculation of

$$result = Product_{New}/Product_{OldMax}$$

is also sound as again it is a multiplication of encrypted values. If a cheating prover wants to show that $result$ decrypts to 1 when $D(result) \neq 1$, they will have to publish an incorrect share that still passes the verifiable decryption proof. The chance of doing this is $1/q$ as shown in Section 5.3.1.

6.5.3 Zero Knowledge

Given the shifted and randomised bid vector New, the alternate form bid vector Alt, the shuffled alternate bid vector $Shuffled_{Alt}$, p, q, g, the public keys $y_1, ..., y_n$, the shares of the decryption for the items in $Shuffled_{Alt}$, and the shares of the decryption for $result$, $z_{1,result}, ..., z_{n,result}$ the zero knowledge property of this proof can be checked by constructing a simulator S that completes the following steps in the random oracle model:

- S outputs the transcript of the simulator for the verifiable shuffle from [17] with the modifications from Section A.5.3 on the input of Alt and $Shuffled_{Alt}$.

- For each item i in $Shuffled_{Alt}$, S outputs the transcript of the simulator from the verifiable threshold El-Gamal shown in Section 5.3.4 on input p, q, g, $Shuffled_{Alt,i}$, I, $z_{1,i}, ..., z_{n,i}$, and $y_1, ..., y_n$.

- S then outputs the transcript of the simulator from the verifiable threshold El-Gamal shown in Section 5.3.4 on input p, q, g, $result$, I, $z_{1,result}, ..., z_{n,result}$, and $y_1, ..., y_n$.

6.6 Example Verifiable Combinatorial Auction

Figure 6.1 illustrates a simple combinatorial auction for two goods with two bidders as used in Section 5.7. There is a group of auctioneers responsible for calculating the auction result at each node other than the start node of the graph. In this example, a $(2, 3)$ threshold scheme is being used. The correct winner of the auction is bidder one with a price of 5 for goods one and two together.

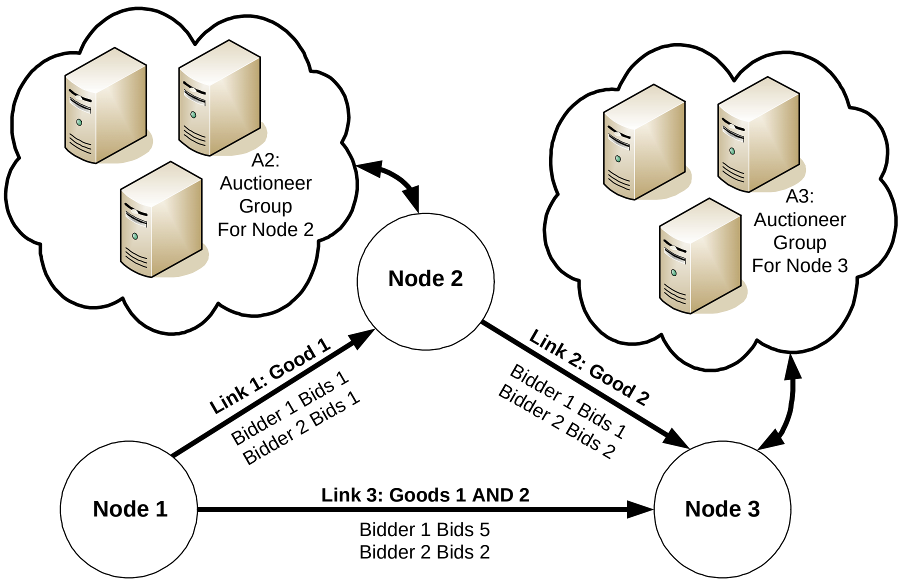


Figure 6.1: Simple Threshold Combinatorial Auction

To perform a public verification that the auction has correctly taken place the following actions are executed in conjunction with the actions executed to calculate the winner of the auction. Any party can verify the proofs that are published to a public bulletin board.

1. Bids are encrypted and published in alternate form for every link using the public key of the node responsible for the link. Bidders publicly publish zero knowledge proofs that every item in the alternate bid vector decrypts to a 1 or a Z. The auctioneers for the node compute the product of the

items in the alternate bid vectors and publish zero knowledge proofs that the products decrypt to Z before converting them to the standard bid vector form.

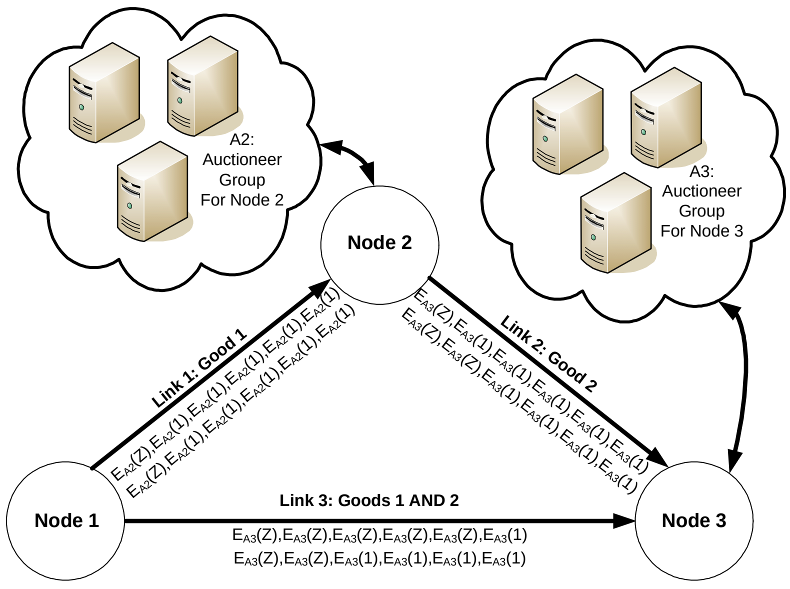

A2:
Auctioneer
Group
For Node 2
A3:
Auctioneer
Group
For Node 3
Node 1
Node 2
Node 3
Link 1: Good 1
$E_{A2}(Z),E_{A2}(1),E_{A2}(1),E_{A2}(1),E_{A2}(1),E_{A2}(1)$
$E_{A2}(Z),E_{A2}(1),E_{A2}(1),E_{A2}(1),E_{A2}(1),E_{A2}(1)$
Link 2: Good 2
$E_{A3}(Z),E_{A3}(1),E_{A3}(1),E_{A3}(1),E_{A3}(1),E_{A3}(1)$
$E_{A3}(Z),E_{A3}(Z),E_{A3}(1),E_{A3}(1),E_{A3}(1),E_{A3}(1)$
Link 3: Goods 1 AND 2
$E_{A3}(Z),E_{A3}(Z),E_{A3}(Z),E_{A3}(Z),E_{A3}(Z),E_{A3}(1)$
$E_{A3}(Z),E_{A3}(Z),E_{A3}(1),E_{A3}(1),E_{A3}(1),E_{A3}(1)$

Figure 6.2: Simple Threshold Combinatorial Auction with Encrypted Bids

2. Figure 6.2 shows the auction graph after the encrypted bids have been submitted and converted in to standard form.

3. The auctioneers in group $A2$ then compute the maximum bid for node 2 and shift and randomises the bids on the outgoing link 2 by the maximum bid amount and publishes zero knowledge proofs that this has been done correctly for every bid that is shifted and randomised.

4. Figure 6.3 shows the auction graph after the bids for link 2 have been shifted and randomised.

5. The auctioneers in group $A3$ then compute the maximum bid m for node 3 which is the optimal path and publish zero knowledge proofs that this was done correctly.

6. The auctioneers in group $A3$ decrypt the bids on the incoming links at position m and publish proofs that this was done correctly to the bulletin board using the technique from Section 5.3.1.

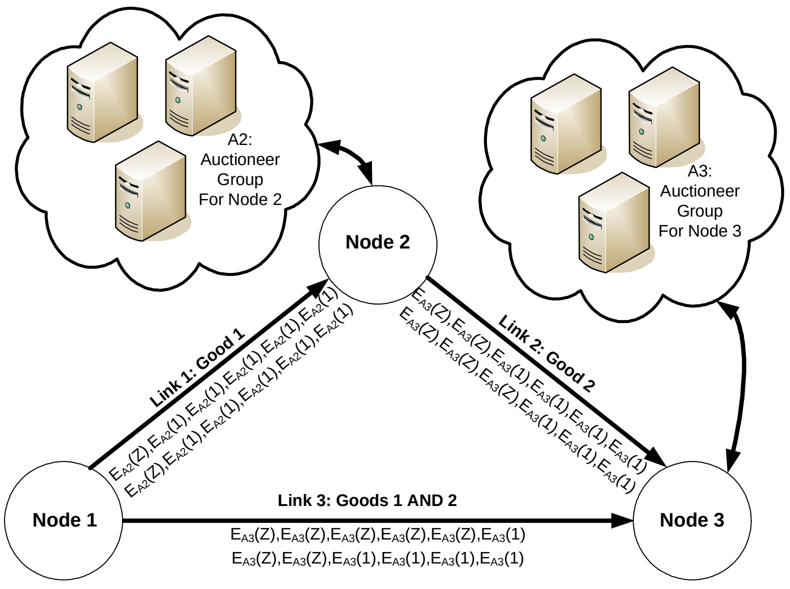

A2: Auctioneer Group For Node 2
A3: Auctioneer Group For Node 3
Node 1
Node 2
Node 3
Link 1: Good 1
E_A2(Z),E_A2(1),E_A2(1),E_A2(1),E_A2(1),E_A2(1)
E_A2(Z),E_A2(1),E_A2(1),E_A2(1),E_A2(1),E_A2(1)
Link 2: Good 2
E_A3(Z),E_A3(Z),E_A3(1),E_A3(1),E_A3(1),E_A3(1)
E_A3(Z),E_A3(Z),E_A3(Z),E_A3(1),E_A3(1),E_A3(1)
Link 3: Goods 1 AND 2
E_A3(Z),E_A3(Z),E_A3(Z),E_A3(Z),E_A3(Z),E_A3(1)
E_A3(Z),E_A3(Z),E_A3(1),E_A3(1),E_A3(1),E_A3(1)

Figure 6.3: Simple Threshold Combinatorial Auction after the Shift and Randomise

6.7 Summary

This chapter presented a public verification protocol for the homomorphic auction protocol. We have restated the threat model and security goals for the public verification protocol, which are similar to those in the group verification protocol. While the group verification protocol relies on less than the threshold t of auctioneers for each group being malicious, the public verification protocol can detect malicious auctioneers even when all the auctioneers in a group are malicious. The public verification presented in this chapter will be computationally expensive. The proofs of the maximum bid and shift and randomise require a verifiable shuffle to be performed on each vector where each shuffle requires $18n$ modular exponentiations where n is the length of the bid vector. The public verification of the auction provides any third party with confidence in the auction result.

Chapter 7

Security Analysis

Throughout this thesis arguments have been made on the completeness, soundness, and zero knowledge of the zero knowledge proofs being used in the public and group verification protocols. These arguments apply when the proofs are used individually but what happens in the auction protocol where different types of proofs are combined and computed multiple times. For example, an auctioneer in the group verification protocol will have to prove for every decryption required in the protocol that their share is correctly computed. Will the combination of all these proofs enable an attacker to extract information about the secret key of the auctioneer?

The original definition of zero knowledge and the simulator presented in Section 3.1 is not zero knowledge under sequential composition. A stronger notion of the simulator for zero knowledge called black-box simulation is zero knowledge under sequential composition. A black-box simulator requires the existence of a universal simulator that given any verifier can simulate the interaction between the prover and the verifier [20]. As our proofs are non-interactive zero knowledge, the transcript involves a message sent from the prover to the verifier containing the proof transcript followed by a one bit output from the verifier specifying whether the proof was accepted or not. The simulators constructed throughout this thesis are black-box simulators as the simulator constructs the proof transcript from the prover to the verifier and can then run the verifier on the transcript to see if it accepts the proof transcript or not. This is possible because the verifier is entirely deterministic and has no random value, otherwise the simulators would need adapting to be black-box simulators. It is worth noting that all known zero knowledge protocols are in fact black-box simulator zero knowledge [21].

7.1 Passive Adversaries

A passive adversarial prover has few options, as by definition they cannot deviate from the protocol. A passive adversarial auctioneer for a particular node in the auction graph will learn the maximum bid value for that node, but this is a limitation of the homomorphic auction protocol and not linked to the verification scheme.

A verifier that is honest but curious may record all the transcripts of the proofs it verifies and use this information to try and extract extra information. Since the zero knowledge proofs presented in this thesis are black-box simulator zero knowledge they are closed under sequential composition. As these proofs are closed under composition, an honest but curious verifier could gain no extra information by recording the transcripts of all the proofs that they have taken part in.

7.2 Active Adversaries

A prover that is an active adversary may try and convince a verifier of the validity of a false assertion. To do this the prover must be able to construct a proof that is not sound. The soundness of the zero knowledge proofs presented has been examined and the chance of a prover being able to prove an invalid statement is $1/q$ where q is a parameter of the El-Gamal encryption used. The smallest key size used is 128 bits long and so $|q| = 128$. This makes the probability of the prover presenting a proof of an invalid assertion $1/2^{128}$. Given that the number of possible keys for the El-Gamal encryption is 2^{128} the possibility that an auctioneer could break the encryption used on the bids and decrypt all the bids on the same link is the same as the probability of an auctioneer being able to present a proof of an invalid assertion that is accepted by a verifier.

When conducting zero knowledge proofs, a verifier that is an active adversary may try and formulate challenges for the prover in such a way that it can learn more than the validity of the assertion being proved. If a verifier was to succeed in learning more information then the proof would not be zero knowledge.

Zero knowledge proofs such as the proof of knowledge of a discrete logarithm and the proof of equality of discrete logarithms have the property of witness extraction. In a proof of knowledge of a discrete logarithm the prover knows a secret value x such that $y = g^x$ where y and g are publicly known. The proof transcript, or the data transferred, consists of three values; a the commitment, c the challenge, and r the response. If the same commitment value is used for two proofs, but a different challenge and response are issued a witness to the knowledge being proved can be extracted. Given two proof transcripts with values a, c_1, r_1 and a, c_2, r_2, and it is known that $g^{r_1} = av^{c_1}$ and $g^{r_2} = av^{c_2}$ dividing one by the other gives:

$$g^{r_1}/g^{r_2} = av^{c_1}/av^{c_1}$$

$$g^{r_1 - r_2} = v^{c_1 - c_2}$$

$$log_g v = r_1 - r_2/c_1 - c_2 = x$$

Clearly, this should be avoided where a verifier can extract the secret value from the prover. As the challenge c is being generated from the random oracle on input a, there is no way for the random oracle to provide two different answers to the same challenge as the cryptographic hash function used is deterministic. This means the proofs should never provide enough information to allow a malicious party to extract a witness to the proof of knowledge.

7.3 Colluding Parties

Suppose some parties in the protocol were colluding to try and subvert the auction protocol. What would be the effect on the verification protocols? There are four main possibilities. A bidder could be colluding with an auctioneer, a prover and verifier could be colluding, a group of auctioneers could be colluding together, and a group of auctioneers could be colluding with a group of bidders.

Suppose a bidder and an auctioneer were colluding to try and accept a bid that was not valid. In the group verification protocol, to be accepted the bid would be required to pass the verification on at least the threshold amount t of auctioneers. This would require t auctioneers to be corrupt and the group verification protocol is only correct as long as less than t auctioneers are malicious. In the public verification protocol the collusion of a bidder and an auctioneer would be of no advantage when trying to get an invalid bid accepted as any other third party can also verify the proof and will see that the bid is invalid.

When using the public verification protocol, a verifier and a prover could collude to accept a proof that is incorrect. For example, suppose there is an auction where there is one verifier that checks the auction that is colluding with one of the auctioneers. If one of the auctioneers did an incorrect shift and randomise and the verifier incorrectly reported that the zero knowledge proof for the shift and randomise was accepted when it was not, any other parties relying on the output verifier would have a misplaced confidence in the auction result. Any party that wants to verify the result should compute the verification themselves and not rely on the output of another verifier as they could be providing false information.

Suppose a group of auctioneers were colluding to try and subvert the auction protocol. In the group verification protocol, it would require a group of at least the size of the threshold of auctioneers to successfully subvert the auction process. A group of auctioneers with less members than the threshold would be detected by the honest auctioneers. In the public verification protocol, a group of auctioneers colluding would still have to publish proofs that would be accepted by a third party verifier that would require them to publish zero knowledge proofs of an incorrect assertion. The chance of being able to prove an incorrect assertion is $1/q$ where q is a parameter of the El-Gamal encryption. The only benefit of having a group of auctioneers colluding would be the increased computational power when doing a brute force attack. If a bidder was also to collude with the group of auctioneers it would have the same effect as the bidder would still need to publish zero knowledge proofs that were accepted by the threshold number of auctioneers in the group verification protocol or, in the public verification protocol, by a third party verifier.

7.4 The Random Oracle Model

This thesis has presented non-interactive zero knowledge proofs that are secure in the random oracle model, but how secure is the random oracle model? This is a question that has received much attention in current research work. On the one hand there is the result that there exist signature and encryption schemes that are secure in the random oracle model but for which any implementation of the random oracle results in an insecure scheme [8]. This implies not just a weakness with the hash function used but a weakness in the random oracle model as the schemes are insecure under any implementation of the hash function. On the other hand, the random oracle model has been used to construct practical secure schemes where there is no currently known exploit [8]. The main conclusion of this work points to the random oracle model being a useful tool for constructing practical proofs that can eliminate a broad range of attacks but it is not a guarantee that no attacks exist.

Chapter 8

Results and Analysis

This chapter presents a complexity analysis of the group and public verification protocols. The details of the implementation of the group verification protocol and the garbled circuits auction protocol are then discussed. Performance results are then presented for the group verification protocol, the original homomorphic auction protocol, and the garbled circuits auction protocol. These results are then analysed and the three auction protocols are compared based on their performance and security properties.

8.1 Complexity

To compare the performance of the public and group verification schemes for the homomorphic auction protocol their complexity can be examined based on the number of exponentiations needed to complete them. This complexity is based on computing the proofs and does not include the verification process. It is assumed that there are 2^g links in the graph where g is the the number of goods, b is the number of bidders, n is the number of auctioneers, and l the maximum bid for a link. Table 8.1 presents the upper bound of the complexity of the two schemes.

As can be seen from the table, the upper bound on the number of modular exponentiations for the public verification is significantly more than for the group verification. This shows that the public verification protocol will be significantly more expensive in terms of both computation and communication than the group verification protocol. The public verification protocol would in all likelihood only be used in auctions of very high value goods such as auctions for radio spectrum.

Proof	Group	Public
Valid Bids	$2^g b(7+2n)$	$2^g b(7+2n)$
Max Bid	$2^g 2ln$	$2^g b(18l+2)$
Shift and Randomise	$2^g 4bl$	$2^g b(20n+2nl)$
Find Winning Bids	$2^g 2bn$	$2^g 2bn$
Total	$2^{g+2}b(7+4n+4l)+2ln$	$2^{g+2}b(9+24n+18l+2ln)$

Table 8.1: Complexity

8.2 Implementation

All implementation was done in Java with the following sections detailing the implementation of the protocols and the testing environment.

8.2.1 Combinatorial Garbled Circuit Auction Protocol

The circuits are implemented as a class with an array of node objects where a node is a representation of a Boolean gate. To execute a circuit, the circuit object loops through the array of node objects until they have all been executed. A node object will only execute if all of it's parent node objects have executed or it is an input node, as described in Section 4.2.3. A circuit object can be created by any party in the garbled circuit auction protocol by using an implementation of the algorithm described in Section 4.3.2 and can be garbled using an implementation of the algorithm described in Section 4.2.2. The VPOT protocol was also implemented which is used by the auctioneer to find the garbled inputs to the garbled circuit. The implementation of VPOT was based on the protocol described in a paper by Juels and Szydlo [28].

The auction issuer, auctioneer, and bidder were then implemented as separate processes. The auctioneer starts the auction and requests a garbled circuit object from the auction issuer passing it the number of bidders, number of goods, and the maximum bid for the auction. The auction issuer creates a garbled circuit object, and returns this along with the output mapping for the circuit to the auctioneer. The auctioneer, auction issuer, and bidder all execute the VPOT protocol to find the garbled input values to the garbled circuit. Finally the auctioneer executes the garbled circuit using the algorithm described above and publishes the output of the garbled circuit when translated by the output mapping. The bidder process is able to spawn multiple threads to represent different bidders. This

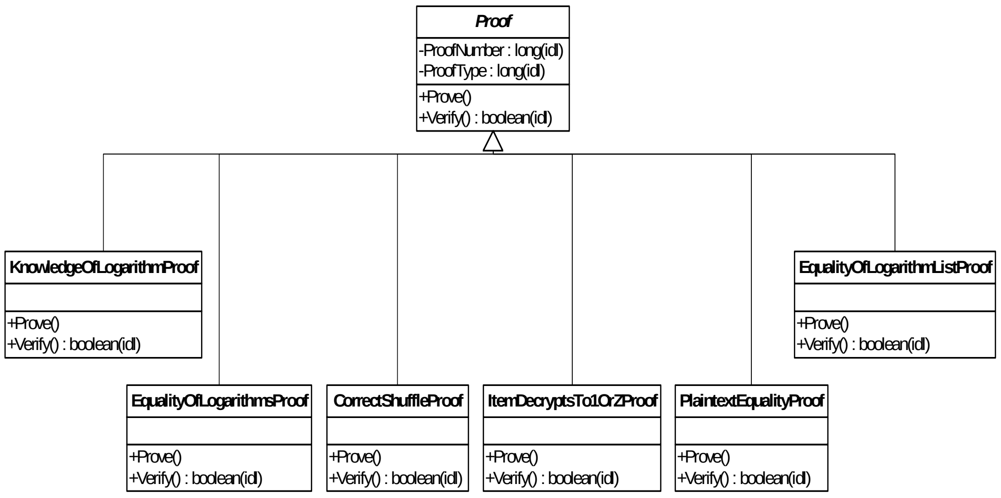


Figure 8.1: UML Diagram of Proofs

makes it be possible to test the auction protocol with a large number of bidders as a thread per bidder has less system requirements than a process per bidder.

As part of his masters thesis Wayne Thomson has designed and implemented a generalised auction framework (GAF) [49]. GAF provides a framework to implement auction protocols and compare their performance. Wayne has ported my implementation of the extended garbled circuit auction protocol to the GAF as a test case for his framework.

8.2.2 Verifiable Homomorphic Auction Protocol

The group verification protocol was first implemented as a prototype that was able to verify the result of a two good homomorphic auction whose result and process were hard coded. To create the prototype, the zero knowledge proofs used in the verification protocols were implemented using an abstract base class called Proof that all the individual proofs extend as shown in Figure 8.1. The implementation of the proofs made extensive use of the Java class BigInteger for the modular arithmetic required for the zero knowledge proofs. The proofs have been implemented as described in Chapter 3 with the random oracle implemented using the secure Java MessageDigest SHA-512 implementation.

After successful testing of the prototype group verification protocol, it needed to be applied to a full implementation of the homomorphic auction protocol in a real system. Wayne has implemented a threshold version of the homomorphic auction protocol in GAF. I have extended Wayne's implementation of the homomorphic auction protocol to implement the group verification protocol.

This involved four main changes:

- Bidder Behaviour: Bidders were extended to publish proofs that every item in the bid vector they have constructed is a 1 or a Z.
- Auctioneers Finding the Optimal Value: Auctioneers were extended to publish zero knowledge proofs that their shares of the decryption of the product of the bids on incoming links are valid. When doing the decryption, auctioneers verify the proofs of the shares before doing the decryption.
- Auctioneers Doing Shift and Randomise: Auctioneers were extended to publish zero knowledge proofs that the shift and randomise operation was done correctly and these are verified by the other auctioneers.
- Auctioneers Finding the Optimal Path: Auctioneers were extended to publish zero knowledge proofs that their shares of the decryption when finding the optimal path are valid. When doing the decryption, auctioneers verify the proofs of the shares before doing the decryption.

8.2.3 Test Environment

With the original homomorphic auction protocol, the group verification protocol, and the garbled circuits auction protocol in the same framework tests can be run to compare the protocols while they are executing in a similar environment with similar communication overheads.

Figure 8.2 shows the testing setup for the garbled circuit auction protocol. The garbled circuit auction protocol has three main components, the auctioneer, the auction issuer, and the bidders. The first test server in the setup ran the GAF component the auction composer that is used to control the auction and run a group of test cases. The second test server ran the auctioneer. The third test server ran the auction issuer. The fourth test server ran the GAF components bid publisher, result publisher, and the auction publisher. These components are used by GAF to signal events in the auction. For example, the auction publisher publishes the result of the auction to any registered listeners. Two bidders were run on each test server and these bidders are able to use multiple threads to simulate multiple bidders if more than eight bidders are required in an auction, my testing went up

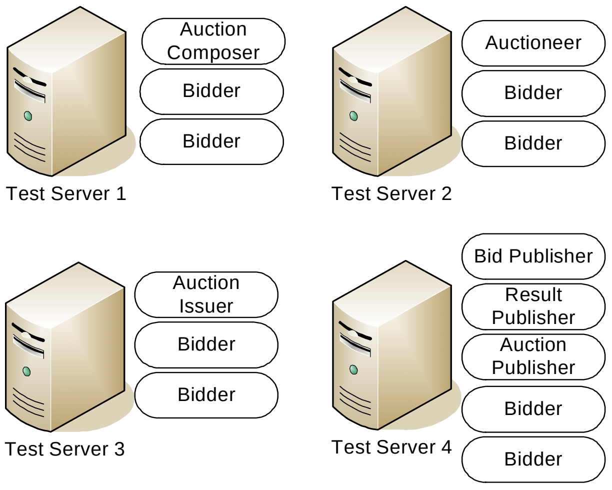


Figure 8.2: Garbled Circuit Auction Protocol Test Setup

to one hundred bidders. Bids in the garbled circuit auction protocol were chosen at random from any value less that the maximum bid. The values of bids does not affect the running time of the garbled circuit auction protocol as the size of the circuit and the auction running time are only affected by the number of bidders, the size of the maximum bid, and the number of goods.

Figure 8.3 shows the testing setup for the homomorphic auction protocol and the group verification protocol. Both of these protocols have several main components, the auctioneer who runs the auction, the evaluators who calculate the result of the auction, and the bidders. The number of evaluators is at least equal to the size of the threshold scheme used. In our test we used a $(2, 3)$ threshold scheme and four evaluators. Only three of these evaluators will be used in any one auction, there is an extra evaluator available in case one fails. The first test server in the setup ran the GAF components the auction composer and the bid publisher as well an auction evaluator. The second test server ran the GAF component bid publisher and an auction evaluator. The third test server ran the auctioneer. The fourth test server ran an auction evaluator. The fifth test server ran the GAF component auction publisher as well as an auction evaluator. Two bidders were run on each test server in the same way as in the garbled circuits auction protocol and these bidders are able to use multiple threads to simulate multiple bidders. The bids in the homomorphic auction protocol and the group verification protocol were chosen differently to the bids in the garbled circuit auction protocol. The bids were chosen to be randomly either one or two. The reason

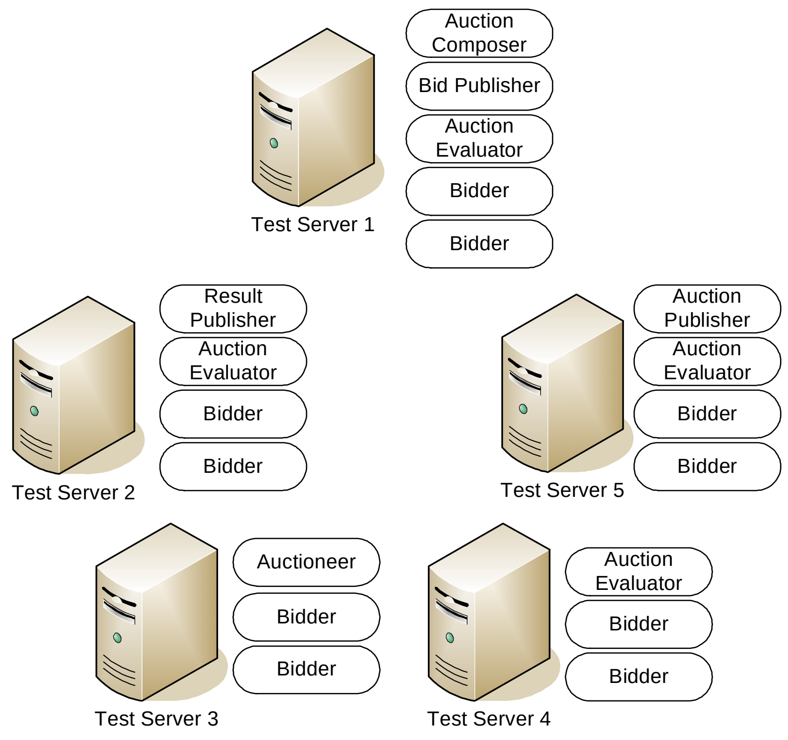


Figure 8.3: Homomorphic Auction and Group Verification Protocol Test Setup

for this is that for both the homomorphic auction protocol and the group verification protocol the execution time of the auction is affected by the values of the bids. When calculating the maximum bid for a link, the product vector is decrypted from right to left. If the maximum bid being decrypted is high, less decryption need to be done to find the value and so this will be quicker than if the bid is low. This is especially true for the group verification protocol where more proofs of equality will need to be calculated and published for low value bids. The low values were chosen so the results show the lower bounds on performance.

8.3 Verification Tests

In some runs of the auction protocol, some of the provers cheat to make sure that the verification schemes detect parties that do not adhere to the auction protocol. These tests check whether the group verification protocol can detect invalid bids, incorrect decryption shares published by auctioneers, and an auctioneer incorrectly doing a shift and randomise. The tests are repeated three times and the number of deviations the verification schemes detect are recorded.

Problem	Detected by Group
Invalid Bid	3 of 3
Invalid Decryption Share	3 of 3
Invalid Shift and Randomise	3 of 3

Table 8.2: Verification Tests

8.4 Verification Performance Results

The computers used for the performance tests were Dell Optiplex GX755s with an Intel Core 2 Duo processor and 2048MB DDR SDRAM. Each test was run thirty times and the average time was taken. A default key size of 128 bits was chosen for the homomorphic auction protocol and the group verification protocol to try and make them comparable, in terms of their privacy preserving properties, to the garbled circuit auction protocol which uses a random function with an output of 128 bits. In the following tests one parameter of the auction protocol is varied and the performance is tested. All other parameters remain the same. The default values for the parameters are:

Parameter	Setting
Number of Bidders	10
Maximum Bid	16
Number of Goods	3
Key Size (for the verification protocols only)	128

Table 8.3: Default Test Parameters

All tests record the total auction time. For the garbled circuit auction protocol this includes the time taken to generate and garble the circuit as well as the time taken to compute the VPOT protocol to find the garbled inputs of the circuit. For the group verification protocol this includes the time taken by bidders to encrypt and submit bids as well as the time taken by the auctioneers to calculate the result of the auction, publish zero knowledge proofs that their actions are correct, and verify the actions of the other auctioneers.

8.4.1 Number of Bidders

Figure 8.4 shows the effect on auction time of increasing the bidders. Increasing the number of bidders causes linear growth in the time taken to compute the

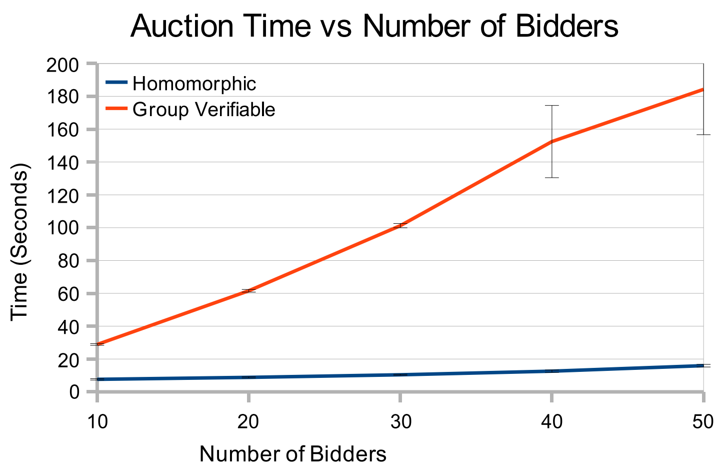


Figure 8.4: Number of Bidders vs Time Taken for Auction

auction. This is because increasing the number of bidders increases the work done to compute the auction protocol linearly. The group verification protocol adds a significant overhead to the original protocol but still has a largely linear growth. The extra overhead is mainly due to the extra proofs and verification that needs to be done to prove all the bidders are submitting valid bid vectors as well as the extra proofs that need to be computed to calculate the optimal path through the graph once the optimal value has been found.

The group verification result for forty bidders seems to show a greater than linear growth. This is caused by the standard deviation of the results for forty and fifty bidders being quite broad. The larger spread of values for forty and fifty bidders could have been caused by other background processes running on the test PCs or by the bidder test PCs having a more variable execution time due to the increased memory demands of the extra proofs and extra threads required for more bidders.

8.4.2 Maximum Price

The effect on auction time of increasing the maximum bid is shown in Figure 8.5. Increasing the maximum bid results in exponential growth of the time taken to compute the auction result. This is true for both the original homomorphic auction protocol and the group verification protocol. The group verification protocol has a slowing exponential growth where the effects of increasing the maximum

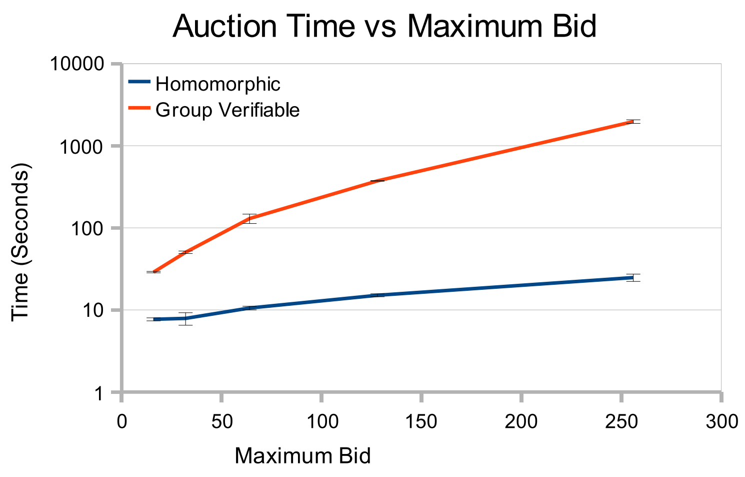


Figure 8.5: Maximum Bid vs Time Taken for Auction

price from 16 to 64 seems to have more effect than increasing it from 64 to 256. It is worth noting that the maximum bid was measured in powers of two so the bid vector lengths tested were 16, 32, 64, 128, and 256 so as to be comparable to the garbled circuit results. Using a linear scale for the tests for the maximum bid would provide more data points to more closely examine the slowing exponential growth.

8.4.3 Key Size

Figure 8.6 shows the effect of increasing the key size used in the El-Gamal encryption on the auction time. The time taken increases exponentially as the key size is increased. This is due to the extra computation requirements of operating on the larger encrypted values. The original homomorphic auction protocol does not seem to increase when using a key size from 128 bits to 384 bits and only seems to experience exponential growth for key sizes greater than 384. A minimum overhead created by the communication costs of the protocol may influence the time taken to compute the auction more than the computational requirements of computing with larger numbers up to a key size of 384. This is also reflected in the group verification protocol where the exponential growth seems to happen for key sizes greater than 256. As there is more computation done on the larger numbers in the group verification protocol it would have an affect on the auction time for smaller key sizes than the original homomorphic auction protocol.

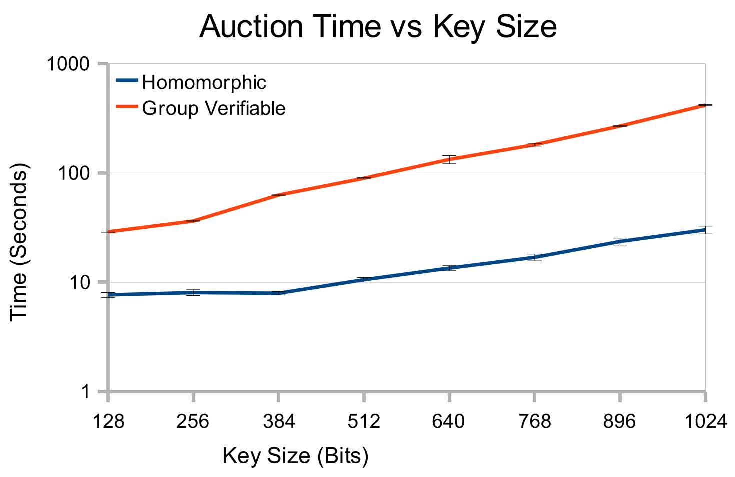


Figure 8.6: Key Size vs Time Taken for Auction

8.4.4 Number of Goods

Figure 8.7 shows the effect of increasing the number of goods on the auction time. Both the original homomorphic auction protocol and the group verification protocol have exponential growth in auction time as goods are increased. This is due to the increase in the number of allocations and hence the auction graph increasing exponentially with the number of goods. With 2 goods there are 4 possible allocations but when the number of goods is increased to 3 there are 8 possible allocations. This is known as the combinatorial auction problem (CAP) which is NP complete and exponential. The original homomorphic auction protocol seems to be exponential with an increasing slope as more goods are added. Although a straight line in this graph would be expected as the goods increase, the increase could be added due to some inefficiency in the original homomorphic implementation such as an inefficient graph creation algorithm that results in extra time taken for each good added. This effect is also shown in the results for the group verification protocol where the time taken grows exponentially with an increase for every good added. As this occurs in the homomorphic auction protocol is does not seem to be a product of the group verification protocol. There is no result for the group verification protocol for five goods as the Java version used was restricted to a maximum of 734MB of memory per process and the evaluators for the group verification protocol require more memory.

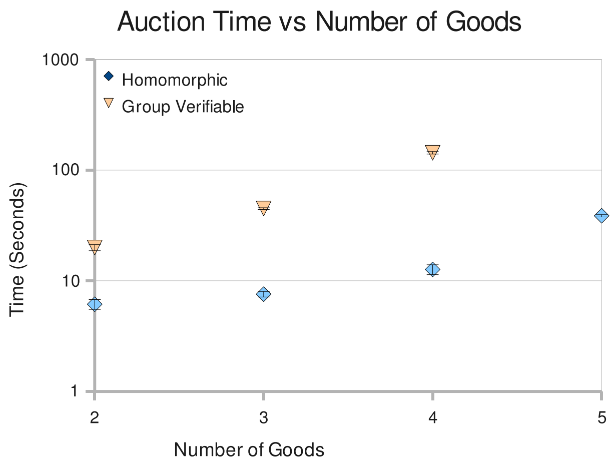


Figure 8.7: Number of Goods vs Time Taken for Auction

8.5 Garbled Circuits Performance Results

8.5.1 The Number of Bidders

The time taken to complete the auction increases linearly as the number of bidders increases as shown in Figure 8.8. This is due to the linear growth in the number of nodes in the circuit needed to compute the auction.

8.5.2 The Maximum Bid

Figure 8.9 shows the time taken to complete the auction when the maximum bid is increased. Increasing the number of bits in the price by one bit increases the maximum price by a power of two.

8.5.3 The Number of Goods

Figure 8.10 shows the time taken to complete the auction increasing exponentially as the number of goods increases.

8.6 Garbled Circuit Size

One of the drawbacks often mentioned about garbled circuits is the size of the garbled circuit that is sent from the auction issuer to the auctioneer. The authors

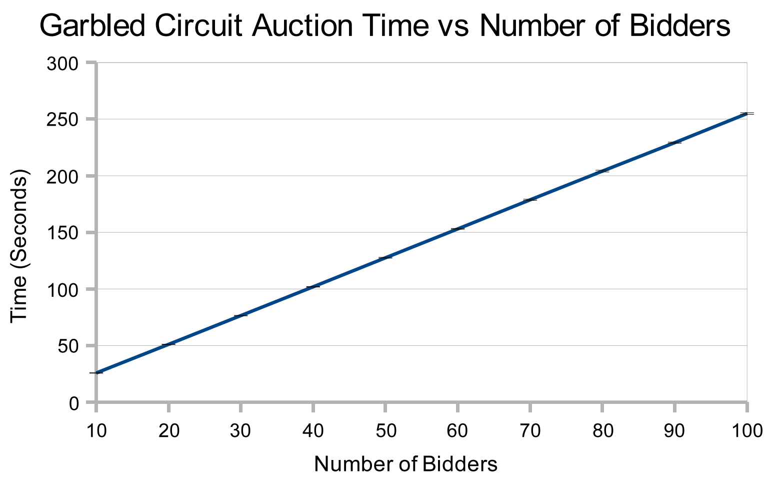


Figure 8.8: Number of Bidders vs Time Taken for Auction

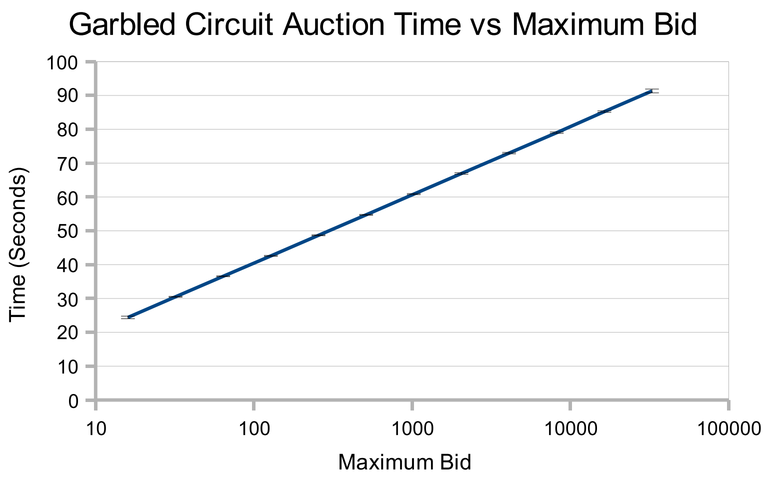


Figure 8.9: Maximum Bid vs Time Taken for Auction

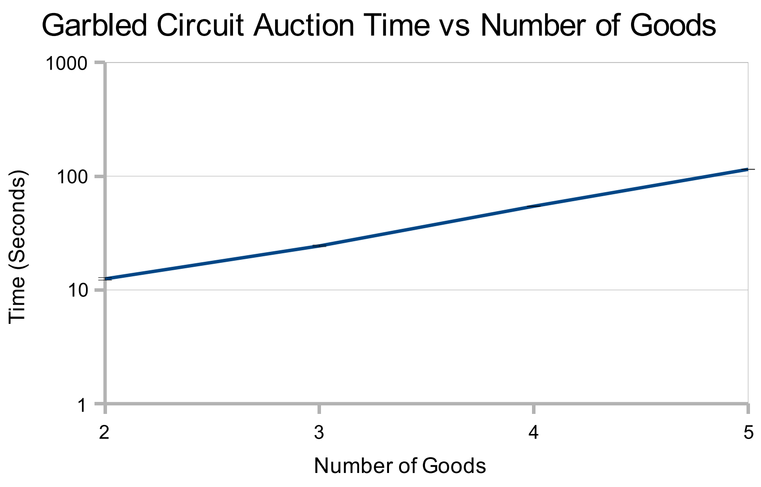


Figure 8.10: Number of Goods vs Time Taken for Auction

of the original paper suggest that the garbled circuits may need to be sent on CD or DVD rather than over the network due to their size [36]. To calculate the size of the auction circuit, the number of two input gates is recorded and multiplied by 4 and then by 128. This is because for every two input gate there are four entries in the gate table and every entry is the size of the output of the random function which in this case is 128 bits. The size of the output mapping is not included in this calculation.

Figure 8.11 shows the size of the garbled circuit increasing linearly as the number of bidders increases. The size of the garbled circuit is proportional to ln(Maximum Bid) as shown in Figure 8.12. Figure 8.13 shows the size of the garbled circuit increasing exponentially as the number of goods increases.

The size of the garbled circuits in these tests would not require a CD or DVD to be sent from the auction issuer to the auctioneer. For example, for an auction with 3 goods, a maximum price of 16, and 100 bidders the size of the garbled circuit is about 6MB. It is worth noting that construction of a more compact combinatorial auction circuit with less nodes would decrease the size of the garbled circuit to be sent.

8.7 Combined Performance Results

Figure 8.14 shows the time taken to compute the auction for the original homomorphic auction protocol, the group verification protocol, and the garbled cir-

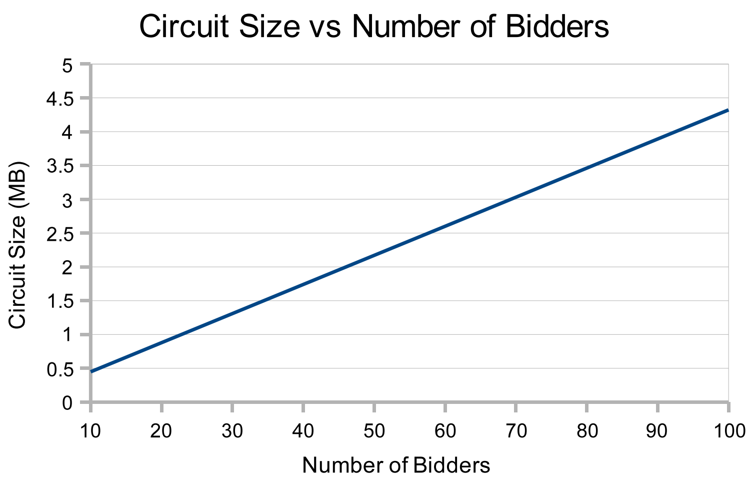


Figure 8.11: Garbled Circuit Size vs Number of Bidders

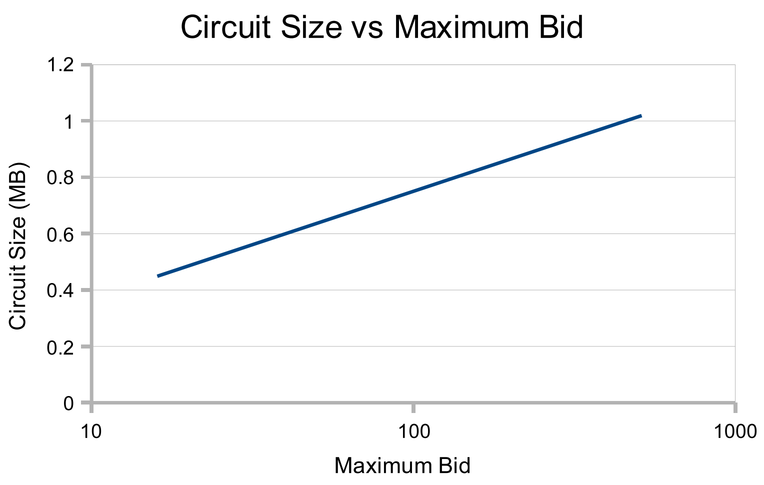


Figure 8.12: Garbled Circuit Size vs Maximum Bid

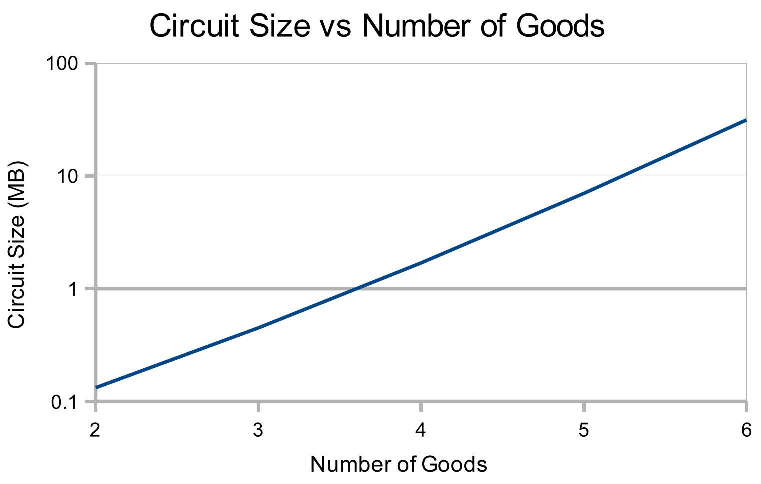


Figure 8.13: Garbled Circuit Size vs Number of Goods

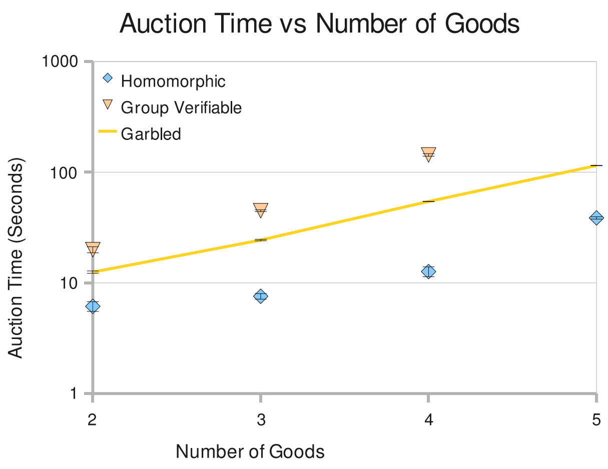


Figure 8.14: Number of Goods vs Time Taken for Auction

cuits auction protocol. The homomorphic auction protocol has the best performance although if the trend of increasing exponential growth continues the garbled circuit auction protocol may be quicker for larger numbers of goods. The group verification protocol has the worst performance of the three auction protocols, but is comparable to garbled circuits especially when computing auctions for two or three goods.

8.8 Analysis of Different Schemes

In this chapter the performance of the original homomorphic auction protocol, the group verification protocol, and the garbled circuit auction protocol have been shown. The group verification protocol takes the longest time of the three protocols and the original homomorphic auction protocol takes the least. Although the group verification protocol adds a significant overhead to the homomorphic auction protocol it also increases the robustness of the protocol by allowing the auction protocol to complete even in the presence of less than t malicious auctioneers. If a malicious auctioneer is a common problem, the group verification protocol may have better performance than the original protocol as the original protocol would either return an incorrect auction result or need to restart if a malicious auctioneer was present. The group verification protocol is also more robust than the garbled circuits protocol where if one of the two parties is malicious or fails the auction would need to be restarted.

The garbled circuit auction protocol has better performance than either the original homomorphic auction protocol or the group verification protocol when a large maximum bid is required. The bid vector notation used in the homomorphic auction protocol causes an exponential increase in the time taken to compute the auction whereas the garbled circuits auction protocol experiences a linear growth in the time taken to compute the auction when the maximum bid is increased by a factor of 2. For auctions where a high bid granularity is required the garbled auction protocol may be the best choice. It is worth noting that the maximum bid for an auction is divided by the number of goods for all three auction protocols. This is because for an auction with say three goods, the longest path through the auction graph would be the allocation of each good individually. If each individual good had the maximum bid value bid for it, the total at the end of the auction would be three times the maximum bid. For this reason the actual maximum bid for an auction is $MaxBid_{actual} = MaxBid/NumberofGoods$.

The garbled circuit auction protocol uses a random function to mask the intermediate values of the circuit and provide privacy. To increase the size of the output of this function would require changes to the protocol and a change of the hash function being used. The homomorphic auction protocol and the group verification protocol can increase the key size used and to protect the privacy of bids in the auction as a parameter. This means that both the homomorphic auction protocol and the group verification protocol are more flexible with their privacy level provided and so for auctions that require a different levels of privacy the homomorphic auction protocol or group verification protocol would be better choices than the garbled circuit auction protocol.

The group verification protocol has stronger verification properties than the garbled circuit protocol. The chance of an active malicious auctioneer being able to force an incorrect auction result undetected by the group verification protocol is $1/q$. For the garbled circuits, the chance of a malicious auction issuer being able to force an incorrect auction result when using the cut and choose verification check that the garbled circuit sent by the auction issuer to the auctioneer is valid is $1/n$ where n is the number of circuits sent by the auction issuer to the auctioneer. For $1/n$ to be equal to $1/q$ the auction issuer would need to send q copies of the garbled circuit to the auctioneer. If each circuit is 6MB and q is a 128 bit number the auction issuer would need to send $6 * 2^{128}$MB of data to the auctioneer which is clearly infeasible. The increased time taken to compute the group verification protocol gives stronger verification properties and so more confidence in the auction result. In auctions where strong verification properties are required the group verification protocol would be the ideal choice. Providing strong verification also encourages more bidders to take part and so can increase the revenue from sellers which would encourage more sellers to take part. If sellers also have confidence in the auction result it can encourage more sellers to take part as the problem of the malicious auctioneer has been minimised.

Chapter 9

Conclusions and Future Work

Online auctions have become a widely accepted way of trading goods and services. One outstanding local example is the New Zealand auction site TradeMe which features over a million listings with more than one and a half million registered users. However, such single good auctions are limited in their ability to express alternatives, compromises or synergy between several goods being auctioned, that is, the net worth of certain goods may increase when combined with other goods. Online combinatorial auctions have been used extensively to allocate truckload transportation and for industrial procurement auctions.

A significant problem with using auctions is the reliance on the trustworthiness of the auctioneer. There is often no means of checking whether the auctioneer has correctly executed the auction without making all bids public, and bid information is potentially commercially sensitive. In current systems this trust is often placed in a central organisation such as TradeMe or the Federal Communications Commission. However, cryptographic techniques can be utilised to provide even stronger guarantees and assurances. The auctioneer can be prevented from breaking privacy guarantees by using a privacy preserving auction where the values of bids are hidden using encryption or obfuscation yet can still be compared to find the winner.

9.1 Contributions and Conclusions

The first contribution of this thesis is a taxonomy reflecting the current state of research into cryptographically secure auctions. In developing this taxonomy it became clear that there was no existing auction scheme that was both verifiable and capable of supporting combinatorial auctions. Two alternative solutions became clear; an existing verifiable auction scheme could be extended to support combinatorial auctions, or an existing combinatorial scheme could be extended to add verifiability. This thesis has explored both of these alternatives.

The second contribution of this thesis was therefore to extend an existing verifiable privacy preserving auction protocol by Naor, Pinkas, and Sumner [36] to conduct combinatorial auctions. A new auction circuit was designed and implemented to extend the garbled circuits auction protocol to support combinatorial auctions. Previously published work had only used the garbled circuit auction protocol to conduct single good (M+1)st price auctions. The construction of the auction circuit involved combining previous work in construction of circuits with some of the techniques used in the homomorphic auction protocol to create an algorithm for constructing a circuit composed of Boolean gates to conduct a combinatorial auction based on the number of bidders, goods, and the maximum bid. These combinatorial garbled circuits have been shown to be a reasonable size (6MB for an auction with 3 goods, a maximum price of 16, and 100 bidders) to send over the network, a previous criticism of the garbled circuit auction protocol.

The third and most significant contribution of this thesis was to add verification to an existing privacy preserving combinatorial auction protocol by Suzuki and Yokoo [53]. In particular, a group and public verification protocol for the homomorphic auction protocol was developed. The verification protocols use zero knowledge proofs to verify the actions taken to compute the auction. The verification protocols have been shown to be secure in the random oracle model. The group verification protocol achieves it's security goals by using zero knowledge proofs to check the actions taken by other parties in the auction protocol. The group verification protocol reveals no information other than that which is revealed by the original auction protocol. It has been shown that it is computationally infeasible for either a bidder or an auctioneer to publish a valid zero knowledge proof of an incorrect action. The chance of a prover being able to cheat the zero knowledge proofs is $1/q$ where q is a parameter of the El-Gamal encryption and $|q| \geq 128$. The public verification protocol achieves it's security goals by giving any third party a mechanism to check the actions taken by parties in the auction protocol. The public verification protocol does reveal what bidder made the highest bid but this can be hidden by using a bidder anonymity scheme. Again, it has been shown that it is computationally infeasible for either a prover to publish a valid zero knowledge proof of an incorrect action with the chance of a cheating prover proving an invalid assertion as $1/q$. The verification protocols make it computationally infeasible for a malicious auctioneer to subvert the auction process preventing the malicious auctioneer and bid filtering. By preventing the malicious auctioneer and bid filtering the verification protocols provide a high degree of confidence in the result of an auction to both bidders and sellers. The verification protocols also increase the robustness of the homomorphic auction protocol by giving auctioneers the ability to detect and ignore invalid decryption shares published by other auctioneers where they would have resulted in an incorrect decryption.

Both the group verification protocol and the garbled circuit auction protocol have been implemented and tested. While the group verification protocol has added a significant overhead to the performance of the original homomorphic auction protocol and is slower than the garbled circuit auction protocol, it has strong verification properties that can give bidders and sellers confidence in the result of the auction protocol. The group verification protocol has stronger verification than the garbled circuit auction protocol where the amount of data needed to be sent to perform the cut and choose verification to the same soundness level would be prohibitive. The group verification protocol can be computed in a reasonable (45 seconds for an auction with 3 goods, a maximum price of 16, and 10 bidders) time and is a practical auction protocol for real world auctions that improves on the security properties of the original homomorphic auction protocol.

9.2 Future Work

9.2.1 Improving Performance of Verification Protocols

The group and public verification protocols presented in this thesis are computationally expensive. This is particularly true of the public verification protocol. Work could be done to find quicker ways to prove the actions taken by the auctioneer, particularly the zero knowledge proof of shift and randomise. If a public verification protocol could be found that did not make such extensive use of the verifiable shuffle of encrypted items, the performance could be greatly increased.

9.2.2 Improved Security Analysis

Although a security analysis has been presented for the zero knowledge proofs used in this thesis, it would be more convincing if the verification protocols could be shown to be provably secure by comparison with known hard problems. Another option to provide greater confidence in the security of the verification protocols would be to run them in a state checker that could check for any possible states that leak information or allow a malicious auctioneer to pass the verification.

9.2.3 Improved Combinatorial Auction Circuit

The combinatorial auction circuit presented in this thesis, while novel and interesting, may be able to be optimised to increase the performance of the garbled circuits auction protocol as well as reducing the amount of data that needs to sent over the network.

Appendix A

Zero Knowledge Proofs

A.1 Proof of Knowledge of a Discrete Logarithm

A.1.1 Completeness

This proof is complete as

$$g^r = g^{z+cx \bmod q} = g^z g^{cx} = av^c$$

It is correct to take $r \bmod q$ as by definition of g, $g^q = 1$ so $g^{q+2} = 1 * g^2 = g^2 = g^{q+2 \bmod q}$.

A.1.2 Soundness

Suppose the prover is cheating and trying to convince the verifier that x satisfies $v = g^x$ when $v \neq g^x$. If a cheating prover chooses r at random and correctly guesses the output of the random oracle c it can set $a = g^r v^{-c}$. If it then outputs the transcript a, r the verifier will accept because $av^c = g^r v^{-c} v^c = g^r$. The chance of a cheating prover correctly guessing the output of the random oracle is $1/q$ as there are q different possibilities for the value a that is the input of the random oracle. A cheating prover could do a brute force over all possible options for the output of the random oracle in at most q steps, which is the same as the amount of work required to break the El-Gamal encryption.

A.1.3 Zero Knowledge

To check the zero knowledge property of this proof, a simulator S can be constructed that completes the following steps on common input p, q, g, and v in the random oracle model:

- S chooses r and c at random and computes $a = g^r v^{-c}$.
- S sets the output of the random oracle on input a to c.
- S outputs transcript a, r.

The verifier then checks that $g^r = av^c$ which holds because

$$av^c = g^r v^{-c} v^c = g^r$$

A.2 Proof of Equality of Discrete Logarithms

A.2.1 Completeness

This proof is complete as

$$g_1^r = g^{z+cx \bmod q} = g_1^z g_1^{cx} = av^c$$

and

$$g_2^r = g_2^{z+cx \bmod q} = g_2^z g_2^{cx} = bw^c$$

A.2.2 Soundness

Suppose the prover is cheating and trying to convince the verifier that x satisfies $v = g_1^x$ and $w = g_2^x$ when $v \neq g_1^x$ or $w \neq g_2^x$. If a cheating prover chooses r at random and correctly guesses the output of the random oracle c it can set $a = g^r v^{-c}$ and $b = g_2^r w^{-c}$. If it then outputs the transcript a, b, r the verifier will accept because $av^c = g_1^r v^{-c} v^c = g_1^r$ and $bw^c = g_2^r w^{-c} w^c = g_2^r$. The chance of a cheating prover correctly guessing the output of the random oracle is $1/q$.

A.2.3 Zero Knowledge

To check the zero knowledge property of this proof, a simulator S can be constructed that completes the following steps on common input p, q, g_1, g_2, w, and v in the random oracle model:

- S chooses r and c at random and computes $a = g_1^r v^{-c}$ and $b = g_2^r w^{-c}$.
- S sets the output of the random oracle on input $a + b$ to c.

- S outputs transcript a, b, r.

The verifier then checks that $g_1^r = av^c$ and $g_2^r = bw^c$ which holds because

$$av^c = g_1^r v^{-c} v^c = g_1^r$$

and

$$bw^c = g_2^r w^{-c} w^c = g_2^r$$

A.3 Proof an Encrypted Item Decrypts to 1 or Z

A.3.1 Completeness

To show completeness the values $c = d_1 + d_2 \bmod q$, $a_1 = g^{r_1} A^{d_1}$, $b_1 = y^{r_1}(B/Z)^{d_1}$, $a_2 = g^{r_2} A^{d_2}$, and $b_2 = y^{r_2} B^{d_2}$ are all checked to hold for both $M = 1$ and $M = Z$.

If $M = 1$:

$$d_2 = c - d_1 \bmod q \text{ so } c = d_1 + d_2 \bmod q$$

$$a_1 = g^{r_1} A^{d_1}$$

$$b_1 = y^{r_1}(B/Z)^{d_1}$$

$$a_2 = g^{r_2} A^{d_2} = g^{w - rd_2} A^{c - d_1} = g^w A^{-(c-d_1)} A^{c-d_1} = g^w$$

$$b_2 = y^{r_2} B^{d_2} = y^{w - rd_2} y^{rd_2} = y^w y^{-rd_2} y^{rd_2} = y^w$$

If $M = Z$:

$$d_2 = c - d_1 \bmod q \text{ so } c = d_1 + d_2 \bmod q$$

$$a_1 = g^{r_1} A^{d_1} = g^{w - rd_1} A^{d_1} = g^w g^{-rd_1} g^{rd_1} = g^w$$

$$b_1 = y^{r_1}(B/Z)^{d_1} = y^{w - rd_1}(B/Z)^{d_1} = y^w y^{-rd_1}(B/Z)^{d_1} = y^w y^{-rd_1} y^{rd_1} = y^w$$

$$a_2 = g^{r_2} A^{d_2}$$

$$b_2 = y^{r_2} B^{d_2}$$

So the proof is complete.

A.3.2 Soundness

Suppose the prover is cheating and trying to convince the verifier that (A, B) decrypts to 1 or Z when it does not. If a cheating prover chooses d_1, r_1, r_2 at random and correctly guesses the output of the random oracle c it can set $d_2 = c - d_1 \bmod q$, $a_1 = g^{r_1}A^{d_1}$, $b_1 = y^{r_1}(B/Z)^{d_1}$, $a_2 = g^{r_2}A^{d_2}$, and $b_2 = y^{r_2}B^{d_2}$. If it then outputs the transcript (A, B), a_1, b_1, a_2, b_2, d_1, d_2, r_1, r_2 the verifier will accept. The chance of a cheating prover correctly guessing the output of the random oracle is $1/q$.

A.3.3 Zero Knowledge

To check the zero knowledge property of this proof, a simulator S can be constructed that completes the following steps on common input p, q, g, y, A, B, and Z in the random oracle model:

- S chooses r_1, r_2, d_1 and d_2 at random.
- S computes $c = d_1 + d_2 \bmod q$.
- S computes $a_1 = g^{r_1}A^{d_1}$, $b_1 = y^{r_1}(B/Z)^{d_1}$, $a_2 = g^{r_2}A^{d_2}$, and $b_2 = y^{r_2}B^{d_2}$.
- S sets the output of the random oracle on input $a_1 + a_2 + b_1 + b_2$ to c.
- S outputs transcript (A, B), a_1, b_1, a_2, b_2, d_1, d_2, r_1, r_2.

The verifier then checks that $c = d_1 + d_2 \bmod q$, $a_1 = g^{r_1}A^{d_1}$, $b_1 = y^{r_1}(B/Z)^{d_1}$, $a_2 = g^{r_2}A^{d_2}$, and $b_2 = y^{r_2}B^{d_2}$.

A.4 Proof of Equality of Two Logarithm Lists

A.4.1 Completeness

This proof is complete as

$$\prod_{i=1}^{n} g_i^{r_i} = \prod_{i=1}^{n} g_i^{z_i + cx_i \bmod q} = \prod_{i=1}^{n} g_i^{z_i} g_i^{cx_i} = v^c \prod_{i=1}^{n} a_i$$

and

$$\prod_{i=1}^{n} m_i^{r_i} = \prod_{i=1}^{n} m_i^{z_i + cx_i \bmod q} = \prod_{i=1}^{n} m_i^{z_i} m_i^{cx_i} = w^c \prod_{i=1}^{n} b_i$$

A.4.2 Soundness

Suppose the prover is cheating and trying to convince the verifier that $x_1, ..., x_n$ satisfies $v = \prod_{i=1}^{n} g_i^{x_i}$ and $w = \prod_{i=1}^{n} m_i^{x_i}$ when $v \neq \prod_{i=1}^{n} g_i^{x_i}$ or $w \neq \prod_{i=1}^{n} m_i^{x_i}$. If a cheating prover chooses $r_1, ...r_n$ at random and correctly guesses the output of the random oracle c it can set $a_1 = g_1^{r_1} v^{-c}$, $b_1 = m_1^{r_1} w^{-c}$, $a_i = m_i^{r_i}$ and $b_i = m_i^{r_i}$ for $i = 2, ..., n$. If it then outputs the transcript $a_1, ..., a_n$, $b_1, ..., b_n$, $r_1, ..., r_n$ the verifier will accept. The chance of a cheating prover correctly guessing the output of the random oracle is $1/q$.

A.4.3 Zero Knowledge

To check the zero knowledge property of this proof, a simulator S can be constructed that completes the following steps on common input p, q, $g_1, ..., g_n$, $m_1, ..., m_n$, w, and v in the random oracle model:

- S chooses $r_1, ..., r_n$ and c at random and computes $a_1 = g_1^{r_1} v^{-c}$ and $b_1 = m_1^{r_1} w^{-c}$.
- S computes $a_i = g_i^{r_i}$ and $b_i = m_i^{r_i}$ for $i = 2, ..., n$.
- S sets the output of the random oracle on input $a_1 + ...a_n + b_1 + ... + b_n$ to c.
- S outputs transcript $a_1, ..., a_n$, $b_1, ..., b_n$, $r_1, ..., r_n$.

The verifier then checks that $\prod_{i=1}^{n} g_i^{r_i} = v^c \prod_{i=1}^{n} a_i$ and $\prod_{i=1}^{n} m_i^{r_i} = w^c \prod_{i=1}^{n} b_i$ which holds because

$$v^c \prod_{i=1}^{n} a_i = v^c v^{-c} \prod_{i=1}^{n} g_i^{r_i} = \prod_{i=1}^{n} g_i^{r_i}$$

and

$$w^c \prod_{i=1}^{n} b_i = w^c w^{-c} \prod_{i=1}^{n} m_i^{r_i} = \prod_{i=1}^{n} m_i^{r_i}$$

A.5 Publicly Verifiable Shuffle of Encrypted Values

A.5.1 Completeness

This poof is complete as stated in [17]. If the prover knows a permutation matrix M_{ij} and integers r_i such that $(AShuffled_i, BShuffled_i) = (g^{r_i} \prod_{j=1}^{n} A_j^{M_{ji}}, y^{r_i} \prod_{j=1}^{n} B_j^{M_{ji}})$

for $i = 1, ..., n$, it is able to provide values that satisfy the 6 equations checked by the verifier.

- Proof 1 is checked by equations 2 and 6.
- Proof 2 is checked by equations 2, 4, and 5.
- Proof 3 is checked by equations 1 and 3, as well as the set of randomly chosen basis.
- Proof 4 is checked using a series of 'Proof of equality of two logarithm lists' proofs. This step is complete due to the completeness of the 'Proof of equality of two logarithm lists' proof.

This proof has been adapted to be non-interactive by changing how the challenge values c_i for $i = 1, ..., n$ are generated. Instead of being generated randomly by the verifier, these values are now generated by inputting the commit values to a hash function. This does not affect the completeness property because the proof is complete for any values of c_i in Z_q which still holds when c_i is generated by a hash function.

A.5.2 Soundness

A cheating prover can convince a verifier that an incorrect shuffle was done correctly by knowing non-trivial integers $a_1, ..., a_n$ such that $\prod_{i=1}^{n} g_i^{a_i} = 1$ or by guessing the output of the hash function. The probability of a cheating prover knowing $a_1, ..., a_n$ is $1/q$. The probability of guessing the output to the hash function is $1/q$. This gives the shuffle protocol a soundness of $1/q$.

A.5.3 Zero Knowledge

The zero knowledge property of this proof is shown in [17]. The proof has been adapted to be non-interactive which requires one modification to the simulator constructed to show zero knowledge in the random oracle model. When the challenge values are chosen randomly from Z_q, the simulator will need to set the output of the hash function to the challenge values on the input of the commit values. This addition gives us a simulator for the verifiable shuffle which shows it is zero knowledge.

The zero knowledge property of the ′Proof of equality of two logarithm lists′ used is shown above by the construction of a simulator.

References

[1] ABE, M., AND SUZUKI, K. M+1-st price auction using homomorphic encryption. In *PKC '02: Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems* (London, UK, 2002), Springer-Verlag, pp. 115–124.

[2] BAUDRON, O., AND STERN, J. Non-interactive private auctions. In *FC'01: Proceedings of the 5th Annual Conference on Financial Cryptography* (February 2001), P. Syverson, Ed., Lecture Notes in Computer Science, Springer-Verlag.

[3] BELLARE, M., AND ROGAWAY, P. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security* (New York, NY, USA, 1993), ACM, pp. 62–73.

[4] BLUM, M., FELDMAN, P., AND MICALI, S. Non-interactive zero-knowledge and its applications. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing* (New York, NY, USA, 1988), ACM Press, pp. 103–112.

[5] BRANDT, F. How to obtain full privacy in auctions. *International Journal of Information Security 5*, 4 (September 2006), 201–216.

[6] BUBENDORFER, K., AND THOMSON, W. Resource Management Using Untrusted Auctioneers in a Grid Economy. In *proceedings of the Second IEEE International Conference on e-Science and Grid Computing (E-SCIENCE)* (Amsterdam, Holland, December 2006).

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

[8] CANETTI, R., GOLDREICH, O., AND HALEVI, S. The random oracle methodology, revisited. *J. ACM 51*, 4 (2004), 557–594.

[9] CHAUM, D., AND PEDERSEN, T. P. Wallet databases with observers. In *CRYPTO '92: Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology* (London, UK, 1993), Springer-Verlag, pp. 89–105.

[10] CHEN, X., LEE, B., AND KIM, K. Receipt-free electronic auction schemes using homomorphic encryption. In *ICISC '03: International Conference on Information Security and Cryptology* (November 2003), pp. 259–273.

[11] CRAMER, R., GENNARO, R., AND SCHOENMAKERS, B. A secure and optimally efficient multi-authority election scheme. In *EUROCRYPT '97: Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology* (London, UK, 1997), vol. 1233, Springer-Verlag, pp. 103–118.

[12] CRAMTON, P., SHOHAM, Y., AND STEINBERG, R., Eds. *Combinatorial Auctions*. MIT Press, 2006.

[13] DAMGARD, I. On sigma-protocols. `http://www.daimi.au.dk/~ivan/Sigma.pdf`.

[14] EBAY. ebay company information. `http://pages.ebay.co.uk/aboutebay/thecompany/companyoverview.html`.

[15] FEDERAL COMMUNICATIONS COMMISSION. About auctions. `http://wireless.fcc.gov/auctions/default.htm`.

[16] FRANKLIN, M., AND REITER, M. The design and implementation of a secure auction service. In *Proceedings IEEE Symposium on Security and Privacy* (Oakland, Ca, May 1995), IEEE Computer Society Press, pp. 2–14.

[17] FURUKAWA, J., AND SAKO, K. An efficient scheme for proving a shuffle. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology* (London, UK, 2001), Springer-Verlag, pp. 368–387.

[18] GAMAL, T. E. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology* (New York, NY, USA, 1985), Springer-Verlag New York, Inc., pp. 10–18.

[19] GOLDREICH, O. Zero-knowledge twenty years after it's invention., 2004.

[20] GOLDREICH, O., AND KRAWCZYK, H. On the composition of zero-knowledge proof systems. *SIAM J. Comput. 25*, 1 (1996), 169–192.

[21] GOLDREICH, O., AND OREN, Y. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology 7*, 1 (1994), 1–32.

[22] GOLDWASSER, S., MICALI, S., AND RACKOFF, C. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing 18*, 1 (1989), 186–208.

[23] GROTH, J. A verifiable secret shuffle of homomorphic encryptions. In *PKC '03: Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography* (London, UK, 2003), Springer-Verlag, pp. 145–160.

[24] GROTH, J. Non-interactive zero-knowledge arguments for voting. In *ACNS '05: Applied Cryptography and Network Security* (2005), pp. 467–482.

[25] HAM, W., KIM, K., AND IMAI, H. Yet another strong sealed-bid auctions. In *SCIS '03: Proceedings of the Symposium on Cryptography and Information Security* (January 2003), pp. 11–16.

[26] HARKAVY, M., TYGAR, J. D., AND KIKUCHI, H. Electronic auctions with private bids. In *WOEC'98: Proceedings of the 3rd conference on USENIX Workshop on Electronic Commerce* (September 1998), pp. 61–74.

[27] I2. i2 website. `http://www.i2.com/`.

[28] JUELS, A., AND SZYDLO, M. A two-server, sealed-bid auction protocol. In *FC '02: Proceedings of the 6th Annual Conference on Financial Cryptography* (2002), pp. 72–86.

[29] JUELS, A., AND SZYDLO, M. A two-server, sealed-bid auction protocol. In *FC '02: Proceedings of the 6th Annual Conference on Financial Cryptography* (2003), Springer-Verlag, pp. 72–86.

[30] KATZ, B. F. *Digital Design: From Gates to Intelligent Machines (Electrical and Computer Engineering Series)*. Charles River Media, Inc., Rockland, MA, USA, 2005.

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

[32] KUROSAWA, K., AND OGATA, W. Bit-slice auction circuit. In *ESORICS '02: Proceedings of the 7th European Symposium on Research in Computer Security* (London, UK, 2002), Springer-Verlag, pp. 24–38.

[33] LIPMAA, H. On diophantine complexity and statistical zero-knowledge arguments. In *ASIACRYPT '03: 9th International Conference on the Theory and Application of Cryptology and Information Security* (London, UK, 2003), Springer-Verlag.

[34] LIPMAA, H., ASOKAN, N., AND NIEMI, V. Secure vickrey auctions without threshold trust. In *FC'02: Proceedings of the 6th Annual Conference on Financial Cryptography* (March 2002), Springer-Verlag, pp. 85–101.

[35] MANHATTEN ASSOCIATES. Manhatten associates website. `http://www.manh.com/`.

[36] NAOR, M., PINKAS, B., AND SUMNER, R. Privacy preserving auctions and mechanism design. In *EC '99: Proceedings of the 1st ACM conference on Electronic commerce* (November 1999), ACM, pp. 129–139.

[37] NEFF, C. A. A verifiable secret shuffle and its application to e-voting. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security* (New York, NY, USA, 2001), ACM Press, pp. 116–125.

[38] NZOUONTA, J., SILAGHI, M.-C., AND YOKOO, M. Secure computation for combinatorial auctions and market exchanges. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems* (Washington, DC, USA, July 2004), IEEE Computer Society, pp. 1398–1399.

[39] NZOUONTA, J. D. An algorithm for clearing combinatorial markets. Master's thesis, Florida Insitute of Technology, 2003.

[40] OFFICE OF COMMUNICATIONS (OFCOM). 2.6 ghz spectrum award consultation. `http://www.ofcom.org.uk/media/mofaq/rcomms/26ghzfaq/`.

[41] PARKES, D. C., RABIN, M. O., SHIEBER, S. M., AND THORPE, C. A. Practical secrecy-preserving, verifiably correct and trustworthy auctions. In *ICEC '06: Proceedings of the 8th international conference on Electronic commerce* (New York, NY, USA, August 2006), ACM Press, pp. 70–81.

[42] PENG, K., BOYD, C., AND DAWSON, E. A multiplicative homomorphic sealed-bid auction based on goldwasser-micali encryption. In *ISC* (September 2005), pp. 374–388.

[43] PENG, K., BOYD, C., DAWSON, E., AND VISWANATHAN, K. Robust, privacy protecting and publicly verifiable sealed-bid auction. In *ICICS '02: Fourth International Conference on Information and Communications Security* (December 2002), pp. 147–159.

[44] PENG, K., BOYD, C., DAWSON, E., AND VISWANATHAN, K. Five sealed-bid auction models. In *ACSW Frontiers '03: Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003* (Adelaide, Australia, 2003), pp. 77 – 86.

[45] SCHNORR, C. P. Efficient identification and signatures for smart cards. In *EUROCRYPT '89: Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology* (New York, NY, USA, 1990), Springer-Verlag New York, Inc., pp. 688–689.

[46] SCHOENMAKERS, B. A simple publicly verifiable secret sharing scheme and its application to electronic. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology* (London, UK, 1999), Springer-Verlag, pp. 148–164.

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

[48] SUZUKI, K., AND YOKOO, M. Secure combinatorial auctions by dynamic programming with polynomial secret sharing. In *Sixth International Financial Cryptography Conference (FC-02)* (March 2002), Springer-Verlag, pp. 44–56.

[49] THOMSON, W. Gaf: A framework for secure combinatorial auctions. Master's thesis, Victoria University Wellington, tba.

[50] TRADEME. Trademe site statistics. `http://www.trademe.co.nz/Community/SiteStats.aspx`.

[51] TREVATHAN, J., GHODOSI, H., AND READ, W. Design issues for electronic auctions. Tech. rep., James Cook University, 2004.

[52] YAO, A. C. Protocols for Secure Computations. In *proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science* (Chicago, IL, USA, 1982), pp. 160–164.

[53] YOKOO, M., AND SUZUKI, K. Secure multi-agent dynamic programming based on homomorphic encryption and its application to combinatorial auctions. In *proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)* (New York, NY, USA, 2002), ACM, pp. 112–119.

[54] YOKOO, M., AND SUZUKI, K. Secure generalized vickrey auction without thirdparty servers. In *proceedings of the 8th International Financial Cryptography Conference (FC-2004)* (Florida, USA, 2004).