An Algorithm for Clearing Combinatorial Markets

by
Josiane Domgang Nzouonta

Diploma of Design Engineer
Electrical Engineering
Universite de Lome
2001

A thesis
submitted to the Graduate School of
Florida Institute of Technology
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Computer Science

Melbourne, Florida
December 2003

We the undersigned committee hereby recommends that the attached document be accepted as fulfilling in part the requirements for the degree of Master of Science in Computer Science.

"An Algorithm for Clearing Combinatorial Markets"
a thesis by Josiane Domgang Nzouonta

Marius C. Silaghi, Ph.D.
Assistant Professor, Computer Science
Major Advisor

Philip J. Bernhard, Ph.D.
Associate Professor, Computer Science
Committee Member

Veton Z. Këpuska, Ph.D.
Associate Professor, Electrical and Computer Engineering
Committee Member

William D. Shoaff, Ph.D.
Associate Professor and Program Chair
Computer Science

ABSTRACT

An Algorithm for Clearing Combinatorial Markets

by

Josiane Domgang Nzouonta

Thesis Advisor: Marius C. Silaghi, Ph.D.

It was recently shown possible to solve single item auctions without revealing any secret except for the solution. Namely, with vMB-share [4], the seller and the buyer only learn each other's identity and the selling price for a chosen $M+1$ pricing scheme. No trusted party was necessary. In this thesis we show how vMB-share can be extended for the clearing of combinatorial negotiations with several items, buyers and sellers. We first show how the more general problem can be reduced to a *virtual form*, relatively similar to the single item auctions. Then, some modifications in the cryptographic techniques of vMB-share are made such that it can offer a solution to problems in *virtual form*. While the problem is known to be NP-complete, a secure multiparty computation that avoids that the secrets leak by measuring computation time necessarily takes an exponential computation cost. Some early experiments show that small realistic negotiations can nevertheless be solved with acceptable effort.

Acknowledgements

First, I would like to thank God, my Father in Heaven, for His Love of me. I believe that I could not have achieved anything without the assurance that He watches upon me every day of my life.

I would like to thank my major advisor Dr. Silaghi for his constant support and guidance through this master thesis. Thank you Dr. Silaghi for your intense devotement to your work. It has been a great source of motivation and inspiration for me. I thank Dr. Bernhard and Dr. Këpuska for taking the time to review this work. Your comments and suggestions were highly appreciated. I thank all the faculties of my department for all their hard work and all the personnel of the web and network services at Florida Tech. I do not want to forget all those who have encouraged and supported me here. Thank you.

Thanks to my father, Jean-Paul and my mother, Jeannette. I also thank all my sisters, and especially Carole. Thank you for your support. I say thanks to Moise and all my friends.

Dedication

To my father, Jean-Paul, and my mother, Jeannette.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Electronic commerce and agent based negotiations still raise a large number of unsolved problems. An illustration is the radio spectrum allocations by the Federal Communications Commission (FCC). The Federal Communications Commission is the US governemental agency in charge of "regulating interstate and international communications by radio, television, wire, satellite and cable" [7]. To allocate radio spectrum, the FCC makes public announcements of available bandwidths. Interested parties submit a pricing proposition for the ranges of their interest. Each bandwidth is attributed to the party offering the highest amount. Certain questions arise regarding the way in which the attribution process is conducted.

1. How can an unsuccesful party be sure that the winning price is indeed higher than her proposition?

2. How can a party express her will to offer more for bundles of bandwidths than for their individual sums.

A formal solution to these questions exists and was recently adopted by the FCC. It consists in running a negotiation process called *auction* in order to sell the bandwidths.

1.1 Auctions and Markets

There exist different kinds of negotiations. Some remarkable instances are auctions and markets.

Definition 1 (Auction:) *An auction designates a public sale in which property or items of merchandise are sold to the highest bidder [8].*

An auction is a process in which all interested parties in a specific item or set of items come together to determine, using specific rules, a party among themselves that wins the good. We distinguish between single item auctions and combinatorial auctions. In a **single item auction**, a unique seller possesses a single item (or multiple units of an item) that he is willing to sell by auctioning it off to multiple agents. When the auction involves a seller (or possibly multiple sellers acting in concert) with items of different types and multiple bidders, the auction is called a **combinatorial auction**. Bidders are allowed to bid on bundle or combinations of resources presented in the auction. A **combinatorial**

market exchange is a generalization of combinatorial auctions allowing for multiple sellers and buyers [18]. In a market, every seller can be a buyer and every buyer is a potential seller.

Some terminology used in the remaining of this report.

Definition 2 (Auctioneer [16]:) *A person who sells by auction or a person whose business it is to dispose of goods or lands by public sale to the highest or best bidder.*

Typically, the auctioneer is either the seller, or a trusted third party. The trusted third party can be a machine (server) or a human (judge).

Definition 3 (Bidder [8]:) *A bidder is a person interested in an item presented at the auction and who is willing to propose a price for the item.*

When dealing with markets, we assume that the market is held by an entity called *market organizer* who enables sellers and buyers to unite. An example of such *market organizer* is the well known Internet-based firm Ebay. A discussion on the relevance of accounting for market organizers in combinatorial market exchange algorithms is presented in [19].

The form in which bids are proposed in an auction helps classify auctions in two groups: sealed-bid auctions and open auctions. *Sealed-bid* auctions keep the bidders' bid values secret and unknown to other participants. Some sealed

bid auction versions are the first-price sealed bid auction and the Vickrey auction [28]. In an *open* auction, everyone learns each other's proposed bids. Two important variants of open auctions are the English auction and the Dutch auction. Our algorithm is currently compatible with first-price sealed bid auctions. A problem we want to address in the future is the extension of our protocol to Vickrey type auctions.

1.2 Problem Statement

As previously defined, a single item type auction involves a single seller with one/multiple units of an item that she presents to the agents engaged in the auction. Protocols have been proposed to successfully conduct such auctions with variable levels of privacy. A novel technique that presents a lot of interesting characteristics is vMB-Share [3]

vMB-Share allows a seller to conduct a single item auction without the need of a third-party as auctioneer. As commonly accepted, it is not possible to completely trust a third-party with the auction result. The risk of an untruthful judge collaborating with a subset of agents or simply modifying the results of the auction for some personal reasons can never be fully excluded. More, the judge can extract information about good reservation prices, removing the auction's truth incentiveness.

vMB-Share prevents such situations by distributing the trust among the seller and the bidders. If the auction consists of $t-1$ agents and one seller, no information about the result can be infered even by any subgroup of $t-1$ participants coming together. Unless all t agents collude, in which case the result of the auction is normally computed, no subgroup of less than t participants can get any information about the result.

vMB-Share is an efficient protocol for single item type auctions. However, single item auctions are not sufficient to correctly model and solve all types of negotiations. We want to extend it to combinatorial market exchanges.

Example: Nancy just got a raise and would like to take a vacation. All flights are booked and the best hotels do not have any availability room during her vacation period. She decides to try and get a fligt ticket and a hotel reservation through an auction website. However, she does not want to acquire the ticket unless she is able to buy a hotel reservation for the same period. Two individuals, Chris and Marc, are each selling one of these items. Marc is not willing to sell his hotel reservation unless he can also sell the tickets he bought for a music concert at the same hotel. This example illustrates a situation which cannot be approached and solved using vMB-Share.

Formerly, situations where a seller is offering sets of distinct items and cases where bidders desire to acquire simultaneously items that do not necessarily

belong to the same seller cannot be solved. There is a need to provide a generalization that will offer a solution to combinatorial market problems, which indeed picture real-life situations more closely.

Some protocols have already been proposed for solving combinatorial auctions which are a special type of combinatorial market exchange. However, we do not know of any that does not require the addition of a third party in the process. We propose a protocol that enables a set of parties to clear a combinatorial market involving multiple resources and sellers with an acceptable level of privacy. No trusted party is needed. The protocol allows sealed-bids and ensure non disclosure of information at any level. Only the seller and the winner of each resource learn the information they need on the winning price. Although constraints can be enforced on the resources by the sellers and the bidders, a winner is to be explicitely determined for each resource separately. A summary of the desired properties of such protocol are:

- Multiple Items: The protocol should allow a seller to auction off multiple items.

- Sellers Constraints: The protocol should enable a seller to add constraints on the goods. He should for example be able to enforce that a subgroup of items be either sold together (not necessarily to the same agent) or not sold at all.

- Bidders Constraints: Each bidder should be given the ability to specify a subset of items that he is willing to buy either completely or not at all.
- Privacy: The trust should be divided among the bidders themselves. No trusted third-party should be involved in the process.
- Correctness: The protocol should be able to get the bidders entries and correctly compute each resource's selling price assuming a first price sealed bid auction.
- Bid Privacy: The only bid to be revealed in the process is the winning bid of the auction. The seller only learns the winner of each resource and the amount he should receive for the resource.

The contribution of this work is a technique that enables a set of parties to compute a first-price sealed bid auction without involving any external party in the process. We designed and programmed an algorithm called *Secure Negotiations Solver (SNS)* which is a generalization of vMB-Share to combinatorial auctions and markets. The protocol is safe against passive attacks (Appendix A).

To achieve this, we combined cryptographic protocols and mathematical concepts and theory. Our protocol ensures a secure and private auction and offers an acceptable level of privacy. Agents involved in the auction interact by exchanging messages. To ensure that a message is comprehensible only

by the addressee, all communications are encrypted using the RSA public key encryption algorithm.

Agents should be able to enforce constraints in their proposed bids. For this reason, we introduce the notion of *auction combination table* which is a table containing all possible attributions of resources to bidders. The privacy is enforced by making the computations on each agent side. This is possible through a conjunction of secret sharing and multiparty computation techniques. We implemented Shamir's secret sharing in this algorithm. The level of privacy of this algorithm is lower than the level of privacy of vMB-Share. This is due to the fact that our implementation uses multiplication of secret values shared with Shamir's scheme.

1.3 Organization of thesis

To help provide a better understanding of the remaining of the thesis, the next chapter focuses on all the concepts and algorithms that we used to construct the algorithm. Chapter 3 presents some protocols proposed for solving single item auctions and combinatorial auctions. vMB-Share is presented in details. In chapter 4, we present and discuss the Secure Negotiations Solver. Chapter 5 presents the theoretical results of the algorithm along with some early experimental results.

Chapter 2

Cryptographic Techniques

We would like to enable a set of agents not necessarily located in a same compound to run a secure combinatorial market exchange. We need secure channels among agents. This is accomplished by encrypting all messages. Bidders' bid privacy is enforced through the usage of secret sharing. Both methods are implemented using modular arithmetic, which characterizes a special case of Galois field. We present each of these mathematical and cryptographic concepts and algorithms in this chapter.

2.1 Galois Field

One desired property of our protocol is its correctness. The computation of the winning bid must be accurate. Therefore, all computations need to output

correct results. Among the operations performed in our protocol, one finds division of integer numbers. As we know, the division of two integers can output an integer result, but can also output a rational number. The set of rational numbers $\mathbb{Q}$ is a field. However, we chose to perform all arithmetic operations in a special field category called Galois field. The algorithm we propose in this work can also be implemented using rational numbers. Before defining the notion of Galois field, we first review the mathematical notion of field in general.

Definition 4 (Field:) *A field F, also denoted (F, +, *) is a set of elements with two binary operations called addition and multiplication such that for all a, b, c in F the following are true: [24]*

- (F, +) is associative, commutative, has an identity element denoted 0 for addition in F, all elements have an additive inverse in F.
- (F, *) is associative, commutative, has an identity element for multiplication in F. Multiplicative inverse: For all a in F, a $\neq$0, there is an element a^{-1} in F such that $a(a^{-1}) = (a^{-1})a = 1$.
- Multiplication is distributive over addition in the field: $a * (b + c) = a * b + a * c$.

In a field, addition, multiplication, subtraction and division can be done without leaving the field. Examples of field are the set of real numbers and the

set of rational numbers. The set of natural numbers is not a field. Computations made in the set of rational numbers are safe against lost of information. However, as we said before, we preferred to perform all the computations in Galois field.

Definition 5 (Galois Field:) *A Galois field (GP) is an algebraic field that has a finite number of members.*

A Galois field is a finite field. The number of element in a finite field determines the order of the field. The order of a finite field must be a power of a prime number p, p^n, where n is a positive integer. The finite field of order p^n is denoted $GF(p^n)$ [24]. A **prime number** is a number that is divisible only by 1 and by itself. When $n = 1$, we have a Galois field of order p. $GF(p)$ is $\mathbb{Z}_p$, namely modular arithmetic modulo p. An example of Galois field, GF (5), is reproduced in table 2.1 along with some arithmetic operations on the field.

2.2 Public Key cryptography

As previously mentioned, agents engaged in the auction need to exchange information among themselves. We want to use secure channels for each pair of agents.

In order to exchange a message between agent A_1 and agent A_2 on a secure channel, we encrypt it. Two encryption schemes are available: symmetric en-

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

(a) Addition modulo 5

x	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

(b) multiplication modulo 5

a	-a	a^{-1}
0	0	–
1	4	1
2	3	3
3	2	2
4	1	4

(c) Inverses modulo 5

Table 2.1: Arithmetic Operations in GF (5)

cryption and public key encryption. In the next sections, we present and discuss both of them after a brief introduction to cryptography.

2.2.1 Introduction to cryptography

Cryptography is the science of secret writing [8]. Cryptography studies show how a readable text, we call it *plaintext*, can be transformed to a form that we call *ciphertext*, so that only the receiver of the message is able to uncover the original text. The first stage of the process, which deals with the transformation of the plaintext, is called *encryption*. The second stage is called *decryption*. Both stages usually involve the use of a key or set of keys. Figure 2.1 shows the flow of a cryptographic process.

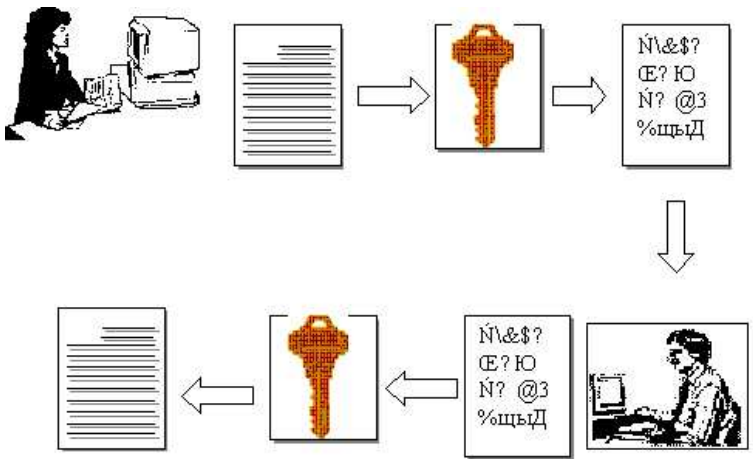


Figure 2.1: Cryptographic process flow

Cryptography is an ancient science. Egyptians were already using non-

standard hieroglyphics to hide messages 1900 BC. Julius Caesar encrypted sensitive messages sent to his troops on the front. He used a now classic encryption scheme which consists of permutations. Each letter of the original message is replaced by the letter occupying the third position after the original letter in the latin alphabet. Figure 2.2 shows some examples of Caesar cipher.

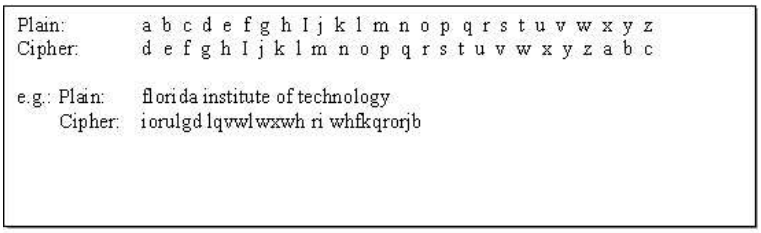


Figure 2.2: Caesar cipher

Replacing each letter by the third one "upward" in the alphabet constitutes the "key" of the Caesar cipher. That key was shared by Caesar and the generals on the front.

The key usage in the encryption and the decryption processes divides cryptographic algorithms in two major groups: If the same key is used to encrypt the plaintext and decrypt the ciphertext, the algorithm is called a private key or symmetric encryption. On the other hand, if two different keys are needed for encrypting and decrypting the data, the algorithm is called public-key encryption. We implement a public key scheme in our algorithm. The following analysis of symmetric encryption explains the reasons of our choice.

2.2.2 Symmetric Encryption

Symmetric or conventional encryption also commonly called private key encryption is the first and most widely spread encryption technique. Here two parties are involved in the process: a sender and a receiver. The sender uses a key and an encryption technique to generate a non intelligible message or *ciphertext*. After receiving the ciphertext, the receiver applies the decryption algorithm and uses *the same key* used at the encryption to uncover the original message. The important element here is the key used in the process. The same algorithm will produce different results for different values of key used.

In symmetric encryption, one key is used for both encrypting and decrypting data. That is the key used by the originator of the message to encrypt the plaintext is the same key that must be used by the recipient to decrypt the ciphertext and uncover the message. The problem here is to get both parties to share the same key. How can the sender securely share the encryption key with the receiver of the message? The safest way is for them to meet personally and perform the exchange. However, our application is aimed at large and distributed audiences that are not necessarily located in the same geographical location. Therefore we cannot consider that option. A secure exchange of the key is not guaranteed.

We will analyze the asymmetric encryption scheme in the next section.

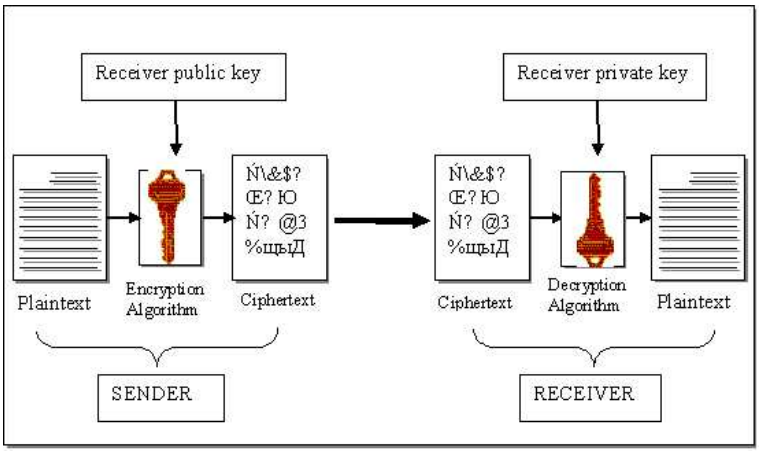
2.2.3 Asymmetric Encryption

Symmetric encryption cannot be implemented to secure communications in our protocol: a secure exchange of keys among agents is difficult. Asymmetric or public key encryption provides a solution to this problem. It was first published by Diffie and Hellman from Stanford University in 1976. It uses two distinct keys called private key and public key respectively.

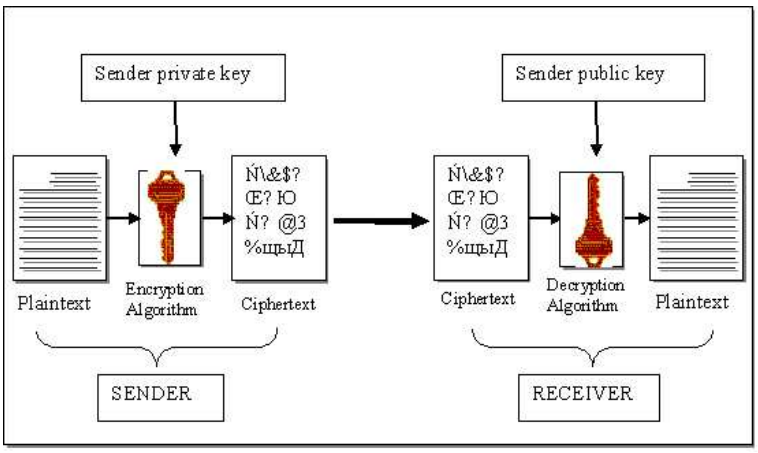
- The public key is publicly known and made available to everybody. It is used for message encryption.
- The private key, which is hidden from everybody and only known by the recipient of the message is used for ciphertext decryption.

Public key encryption is an asymmetric process: the parties involved in the process are not equal. The originator of the message can encrypt the message, but he cannot decrypt the message. The decryption process requires the use of the private key which is kept secret by the receiver.

Figure 2.3 shows the use of public key cryptography (a) for encrypting/decrypting data and (b) for authentication purpose.



(a) Asymmetric encryption process



(b) Authentication process

Figure 2.3: Public key cryptography

2.2.4 Public Key Encryption Algorithms

In this section, we present two major public key encryption algorithms: the RSA algorithm and the El-Gamal algorithm. Our protocol implements the RSA algorithm.

RSA Encryption

Rivest, Shamir and Adleman introduced the RSA algorithm in 1977. The acronym RSA is given after their names. RSA is the most widely used public key algorithm.

RSA is based on exponentiation modulo a prime in a Galois field. Its security relies on the difficulty to find the discrete logarithm of a number modulo a prime [24].

Algorithm Description: To communicate with other participants, each agent needs to generate a pair of encryption keys. The keys are used to encrypt and decrypt messages. The following summarizes all the computation steps required for encrypting or decrypting a plaintext.

- Keys generation

    1. Randomly select two large prime numbers p and q
    2. Compute their product $n = p * q$ and the Euler Totient function of n, $\phi(n) = (p-1) * (q-1)$
    3. Randomly select an encryption key e such that $1 < e < \phi(n)$ and $gcd(e, \phi(n)) = 1$, where $gcd(e, \phi(n))$ designates the greatest commong divisor of e and $\phi(n)$
    4. Find d such that $e * d = 1 \ mod \ \phi(n)$ and $0 \leq d \leq n$

5. Publish public key $K_E = \{e, n\}$ and keep secret key $K_D = \{d, p, q\}$

- Encryption of message M

  1. Get recipient's $K_E = \{e, n\}$
  2. Compute $C = M^e \ mod \ n$, where $0 \leq M \leq N$

- Decryption of ciphertext C

  1. Take $K_D = \{d, p, q\}$
  2. Compute $M = C^d \ mod \ n$

Note that the key generation is made once. All encryption and decryption operations use the same pair of keys. Another encryption algorithm that is also widely used is the El-Gamal algorithm.

El-Gamal

El-Gamal algorithm is also based on exponentiation modulo a prime in a Galois field. As RSA, its security relies on the difficulty of factoring large numbers. El-Gamal derives from the Diffie-Hellman key exchange scheme. The algorithm is presented below.

Algorithm Description: In El-Gamal algorithm, agents also create pairs of keys for use when encrypting/decrypting messages.

- Keys generation

    1. Randomly select a large prime number p and two large numbers g and x, such that $g < p$, $x < p$; g is called the generator.
    2. Compute $y = g^x \ mod \ p$
    3. Publish public key $K_E = \{y, g, p\}$ and keep secret key $K_D = \{x, g, p\}$

- Encryption of message M

    1. Randomly select a number k relatively prime to p-1
    2. Compute $a = g^k \ mod \ p$ and $b = y^k * M \ mod \ p$; C = a,b

- Decryption of ciphertext C

    1. Compute $M = b/a^x \ mod \ p$

Here also the keys are generated only once. All subsequent encryption and decryption operations use the same pair of keys.

The encryption algorithm transforms messages to an unintelligible form. This contributes in securing the communications. We now present how the data is generated before being encrypted and transmitted to other agents.

2.3 Multiparty Computations

Our protocol computes some addition and multiplication of the values of the proposed bids. These operations are performed by each agent separately. However, an agent should not learn other agents bids. Secret sharing concepts and multiparty computation techniques enable the computations to be made on shares of secrets which by themselves reveal no information on the values of the secrets.

2.3.1 Secret Sharing

Motivation: Let's suppose that Alice and Bob came across a map to a hidden treasure. They would like to go home and get ready for this exciting trip. Now, who is going to keep the map? Suppose Alice and Bob do not really trust each other and are afraid that if the other one gets the map, he/she might go alone and get all the treasure for himself/herself. They need a way to share the map so that no one would be left out of the trip. How can this be done? [11]

In more general terms, how can a secret s be shared among n agents so that a regroupment of t agents out of the n are easily able to recover the secret but no information about s is revealed from the complete knowledge of $t-1$ shares out of the t shares?

This situation illustrates a typical problem which can be solved by apply-

ing secret sharing techniques. Multiple secret sharing schemes are available for cryptographic algorithms. We present here Blakley's and Shamir's secret sharing schemes.

2.3.2 Blakley's Threshold Secret Sharing Scheme

In Blakley's (t, n) threshold scheme [2], a secret s is represented as a t-dimension point. Each participant receives as share the equation of a $t-1$ dimensional plan that contains s.

In order to recover the secret, t participants are needed. If a subgroup of $t-k$ agents collude, they are able to determine only that the secret belongs to a *k-dimensional* plane. The complexity of this algorithm motivated our choice of Shamir's secret sharing scheme for our algorithm.

2.3.3 Shamir's Secret Sharing Scheme

Shamir's secret scheme [21] is based on polynomial interpolation of pairs of points (x_i, y_i) on a polynomial $f(x)$.

Let $f(x)$ be a $t-1$ degree polynomial: $f(x) = s + a_1x + a_2x^2 + ... + a_{t-1}x^{t-1}$ where $a_0 =$ s and a_i are random numbers.

The secret s, which correspond to the value of the polynomial in $x = 0$, is shared by computing t distinct pairs (x_i, y_i) on the polynomials with $x_i \neq 0$. Each agent A_i is assigned the pair (x_i, y_i). A technique where any t out of n

agents can retrieve a secret while $t-1$ cannot find anything is called a (t, n) **threshold scheme**.

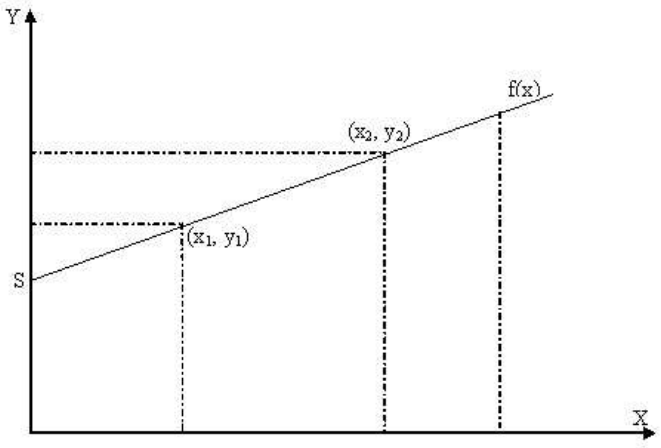


Figure 2.4: Polynomial Interpolation

To reconstruct the secret, the participants jointly, by Lagrange interpolation, compute the value of the polynomial in $x = 0$ using the following formula:

$$s = f(0) = \sum_{i=1}^{n} y_i f(i) \quad where \quad y_i = \prod_{j=1, j\neq i}^{n} \frac{j}{j-i}$$

Any t pairs of (x_i, y_i) uniquely determine s but any subgroup of (t-1) pairs provides absolutely no information about s.

Numerical Sample: Suppose that a secret $s = 18$ needs to be shared among five agents with a (3, 5) threshold scheme (t=3 and n=5). We perform computations in $\mathbb{Z}_{19}$; The sharing is a (3,5) sharing scheme; therefore, the polynomial function f(x) must be of power $3 - 1 = 2$.

$f(x) = s + a_1x + a_2x^2 = 18 + 16x + 9x^2$

The coefficients a_1 and a_2 are randomly selected. Each participant receives a point on the polynomial (participant i receives $f(i)$); This is his share of the secret.

$f(1) = 18 + 16 * 1 + 9 * 1^2 = 5$

$f(2) = 18 + 16 * 2 + 9 * 2^2 = 10$

$f(3) = 18 + 16 * 3 + 9 * 3^2 = 14$

$f(4) = 18 + 16 * 4 + 9 * 4^2 = 17$

$f(5) = 18 + 16 * 5 + 9 * 5^2 = 0$

Three agents are needed to recover the secret value which is the value of the polynomial in $x = 0$. Any combination of two agents cannot infer anything on the secret. If three participants (1, 2 and 3 for example) come together to uncover the secret value, they can solve the system of equations formed by their shares (f(1), f(2) and f(3)) or apply Lagrange interpolation on these values. User 1, 2 and 3 each has a share value of 5, 10, 14 respectively.

Lagrange Interpolation:

$y_1 = 3$; $y_2 =$ -3; $y_3 = 1$;

$a_0 =$ 3*5 + (-3)*10 + 1*14 = -1 equiv 18 in $\mathbb{Z}_{19}$

System of equations:

$$\begin{cases} f(1) = a_0 + a_1 + a_2 = 5 \\ f(2) = a_0 + 2 * a_1 + 4 * a_2 = 10 \\ f(3) = a_0 + 3 * a_1 + 9 * a_2 = 14 \end{cases}$$

The resolution of this system of equations outputs a_0=18, a_1=16 and a_2=9 in $\mathbb{Z}_{19}$.

Shamir's secret scheme is a fully secure scheme. However, we need not only prevent subsets of agents from accessing any information other that what they are supposed to know, but we also need to verify that the values sent are correct.

2.3.4 Verifiable Secret Sharing Schemes

Agents that do not follow the protocol may destroy the results or be able to infer the value of other agents' bids. We must ensure that each agent complies with the protocol. In the computation of a bid's shares for example, it is important to check that the share value y_i sent to agent A_i truly equals $f(x_i)$. Some verifiable secret sharing schemes have been proposed, that enhance Shamir's sharing scheme by adding public verification of the values of shares. Although publicly performed, the verification is completely secure and does not disclose any information other that what was already known.

Algorithm Description [10] To enable verification of each computed values, each agent creates an additional polynomial of degree $t-1$.

$v(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{t-1} x^{t-1}$ where all a_i are random numbers.

Each agent publishes $S = g^s$ and $F_j = g^{a_j}$ for all j, $1 \leq j \leq n$. For each share $y_i = f(x_i)$ computed, the agent publishes $Y_i = g^{y_i}$. The verification of the correctness of the value of y_i sent to agent A_i is made by confirming that

$$Y_i = g^{y_i}$$

where $Y_i = S. \prod_{j=1}^{t-1} F_j^{(x_i^j)} \ (mod\ p)$.

2.3.5 Multiparty Computations

In our protocol, the computations of addition and multiplication of bids are made only by agents, without a trusted party. This is achieved by using multiparty computation (MPC) techniques [29] that enable distributed but yet secure computations of the secret values of bids.

Definition 6 (Secure Multiparty Computation [9]:) *Secure multiparty computation allows two or more parties to evaluate some function of their inputs, such that no more is revealed to a party or a set of parties about other parties' inputs and outputs, except what is implied by their own inputs and outputs.*

The result of the function is known to everyone in the group, but no one learns anything about the inputs of any other members other than what is obvious from the output function.

Motivation Suppose that the members of the executive commitee of some student association are sitting together for a chapter meeting. At a certain point of time, they express the need to know the average value of their paychecks earnings. However, no one is willing to reveal his bi-weekly student wage amount to the other participants. We'll assume that our virtual commitee contains four (4) students, Alice, John, Nancy and Chris. Below we present a very simple and basic protocol for computing the average amount of the paycheck of a commitee member.

2.3.6 Sample MPC Protocol

The following is a very simple protocol that can be used to solve the above exposed problem. [20]

1. Alice adds a secret random value to her salary and, encrypts the result with Chris's public key, and sends it to Chris.

2. Chris decrypts Alice's result with his private key. He adds his earning to what he received from Alice, encrypts the result with John's public key, and sends it to John.

3. John decrypts Chris's result with his private key. He adds his paycheck amount to what he received from Chris, encrypts the result with Nancy's public key, and sends it to Nancy.

4. Nancy decrypts John's result with her private key. She adds her earning to what she received from John, encrypts the result with Alice's public key, and sends it to Alice.

5. Alice decrypts Nancy's result with her private key. She subtracts the random number from step 1 to recover the sum of everyone's earnings.

6. Alice divides the result by the number of people (four in the case) and announces the result.

This sample protocol presents some weaknesses. It assumes that all members in the commitee are truthful. At any stage of the computation, a member could have input an amount different from his real earning. The output average result would have been wrong. More, any of the members could change the amount of the previously summed values by just changing the value he received from his predecessor. A greater risk lies in the fact the power is not equally distributed to all participants. Alice gets to know the result first and can therefore modify it without other participants' knowledge. She could also subtract in step 5 a value different from the one she input in step 1. No other participants could perceive it. Alice could be prevented from doing this by requiring her to commit

to the random number she used in step 1 using a commitment scheme, but when she will reveal the random number at the end of the protocol, Chris will learn her paycheck amount.

To solve this problem, another solution is to use an association of secret sharing scheme and multiparty computation techniques. Each paycheck amount becomes a secret shared among all participants. More specifically, each member of the commitee computes the shares of his earnings using a (4, 4) threshold Shamir's secret sharing scheme and send each share to a member. After receiving shares from other members, each student sums all the values and divides the obtained sum by the number of members present, 4. To uncover the average value, the members jointly compute, by Lagrange interpolation, the constant term of the polynomial defined by their shares. This term corresponds to the average value of their earnings. The average function computation is distributed to all members and each of them is individually needed to determine the result of the computation: the power is equally distributed.

In the next sections of this chapter, we present the mathematical foundations behind this technique. These algorithms and concepts are the same that enable our algorithm to allow a complete unfolding of the auction without revealing any of the involved details.

2.3.7 Addition of Secrets

Let's consider the distribution of two secrets s_1 and s_2 using Shamir's secret sharing scheme (Section 2.3.3) with a (t, n) threshold. The polynomials used are of the form:

$$\text{f(x)} = s_1 + a_1 x + a_2 x^2 + ... + a_{t-1} x^{t-1}$$

$$\text{g(x)} = s_2 + b_1 x + b_2 x^2 + ... + b_{t-1} x^{t-1}$$

Each agent A_i receives the shares f(x_i) and g(x_i). If each agent A_i adds the shares he has, f (x_i) + g (x_i), he obtains h (x_i). $(x_i, h(x_i))$ is a point on the function representing the polynomial h(x) = f(x) + g(x). The new secret value h(x_i) of each agent A_i define a sharing of the secret s = $s_1 + s_2$ with a (t, n) threshold scheme:

$$h(x) = (s_1 + s_2) + (a_1 + b_1)x + ... + (a_{t-1} + b_{t-1})x^{t-1}$$

Numerical Sample: Three friends share two secrets $s_1 = 18$ and $s_2 = 5$ using a (3, 5) threshold scheme (t=3 and n=5). Computations are performed in $\mathbb{Z}_{19}$; User 1, 2 and 3 each received two shares of value {5,10}, {10,0} and {14,13} corresponding to the secrets s_1 and s_2 respectively. They use a (3,5) threshold sharing scheme; therefore, the polynomials $f(x)$ and $g(x)$ used to share s_1 and s_2 must be of power $3 - 1 = 2$. The sum of two polynomials of degree 2 is a polynomial of degree 2. Each participant adds together her shares to obtain a point on $h(x) = f(x) + g(x)$.

$h(x) = a_0 + a_1x + a_2x^2$

We have: h(1) = 5+10 = 15; h(2) = 10+0 = 10; h(3) = 14 + 13 = 8 in $\mathbb{Z}_{19}$

To compute the value of the polynomial h(x) in $x = 0$, we can resolve the system of equation formed by h(1), h(2) and h(3) or apply Lagrange interpolation on these coefficients.

Lagrange Interpolation:

$y_1 = 3$; y_2 = -3; $y_3 = 1$;

a_0 = 3*15 + (-3)*10 + 1*8 = 4 in $\mathbb{Z}_{19}$

System of equations:

$$\begin{cases} h(1) = a_0 + a_1 + a_2 = 15 \\ h(2) = a_0 + 2 * a_1 + 4 * a_2 = 10 \\ h(3) = a_0 + 3 * a_1 + 9 * a_2 = 8 \end{cases}$$

The resolution of this system of equations gives a_0=5, a_1=3 and a_2=2 in $\mathbb{Z}_{19}$. Verification: 18 + 5 = 23 equiv 4 in $\mathbb{Z}_{19}$.

2.3.8 Multiplication of Secrets

In this subsection, except when the multiplication involves the value 0, the term *multiplication* can always be replaced by *division*.

The multiplication of a shared secret with a publicly known constant value c is straight forward. It basically consists of performing the multiplication on

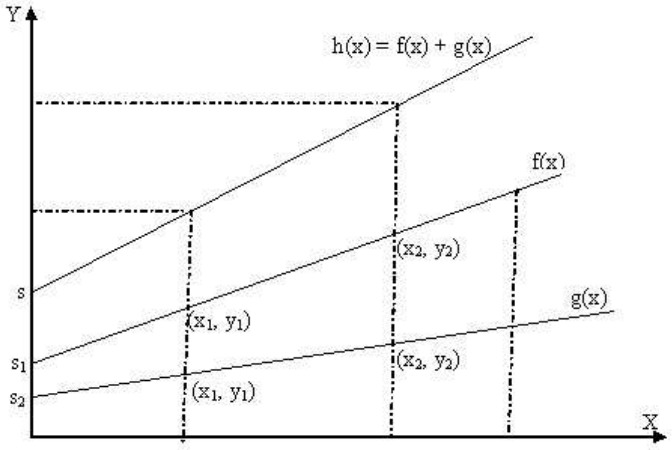


Figure 2.5: Addition of secrets

each bidder side. The shares obtained from that operation are valid shares of c* f (x).

Let's consider a secret s_1 shared among n agents using Shamir's secret sharing scheme with threshold (t, n). $f(x) = s_1 + a_1x + a_2x^2 + ... + a_{t-1}x^{t-1}$. Furthermore, let's suppose that each agent A_i received the share $f(x_i)$. To obtain the value of $c * s_1$, each agent will compute the multiplication $h(x_i) = c * f(x_i)$ on his side. These new shares define a sharing of the secret $s = c * s_1$ with a (t, n) threshold scheme and the associated polynomial function is

$h(x) = c * s_1 + c * a_1x + c * a_2x^2 + ... + c * a_{t-1}x^{t-1}$

The multiplication of two shared secrets is more involved. Let's suppose we add a second secret s_2 to the secret s_1 above. Secret s_2 is shared among the

participants using a (t, n) threshold with the polynomial

$g(x) = s_2 + b_1x + b_2x^2 + ... + b_{t-1}x^{t-1}$.

Each agent now has two shares corresponding to the two secrets values. If we multiply the two shares together, we do obtain a point on the function representing the polynomial h (x) = f (x) * g (x). However, this polynomial is the result of the multiplication of two polynomials of degree $t - 1$. The degree of h(x) is therefore $2t - 2$. This correspond to a $(2t - 2, n)$ threshold scheme. To uncover the secret value, we need $2t - 1$ agents which is twice as much as currently needed. If $2t - 2 > n$ we will even miss some data. This is the reason why a $(\lfloor \frac{n+1}{2} \rfloor, n)$ threshold scheme is used to share secrets whenever multiplication of secrets is to be performed.

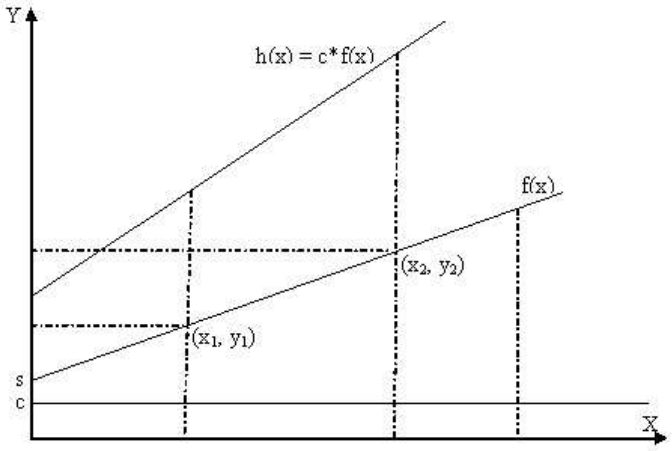


Figure 2.6: Multiplication of secrets

Once we have obtained the points on h(x), we are able to uncover the secret by involving all the n agents in the computation.

Numerical Sample: Two secrets $s_1 = 18$ and $s_2 = 5$ are shared among two participants with a (2, 3) threshold scheme (t=2 and n=3). We perform computations in $\mathbb{Z}_{19}$; User 1, 2 and 3 each received two shares of value {1,8}, {3,11} and {5,14} for s_1 and s_2 respectively. The sharing scheme is a (2,3) threshold sharing scheme; therefore, the polynomials $f(x)$ and $g(x)$ used to share s_1 and s_2 must be of power $2 - 1 = 1$. The product of two polynomials of degree 1 is a polynomial of degree 2. Each participant multiplies her shares together to obtain a point on $h(x) = f(x) * g(x)$.

$h(x) = a_0 + a_1x + a_2x^2$ We have: h(1) = 1*8 = 8; h(2) = 3*11 = 14; h(3) = 5*14 = 13 in $\mathbb{Z}_{19}$

To compute the value of the polynomial h(x) in $x = 0$, we can resolve the system of equation formed by h(1), h(2) and h(3) or apply Lagrange interpolation on these coefficients.

Lagrange Interpolation:

$y_1 = 3$; y_2 = -3; $y_3 = 1$;

a_0 = 3*8 + (-3)*14 + 1*13 = -5 equiv 14 in $\mathbb{Z}_{19}$

System of equations:

$$\begin{cases} h(1) = a_0 + a_1 + a_2 = 8 \\ h(2) = a_0 + 2 * a_1 + 4 * a_2 = 14 \\ h(3) = a_0 + 3 * a_1 + 9 * a_2 = 13 \end{cases}$$

The resolution of this system of equations gives a_0=14, a_1=7 and a_2=6 in $\mathbb{Z}_{19}$. Verification: 18 * 5 = 40 equiv 14 in $\mathbb{Z}_{19}$.

2.4 Conclusion

In this chapter, we have presented the mathematical concepts and cryptographic algorithms that we used in our work. To enable an auction without involving third party, the computations are distributed among agents. Two important aspects associated with this dispersion are:

- Securing the communications among agents: Because the computations are distributed, some interactions among agents will take place. The interactions need to be secured. In this chapter, we have shown that this is securely achieved through encryption of messages. An analysis of symmetric encryption and asymmetric encryption was presented. This analysis clarified the reasons of our choice of public key encryption as our encryption scheme.

- Ensuring that no information is revealed: We presented secret sharing

and multiparty computation techniques which enable our protocol to be concurrently run by all agents, removing the need of a centralized computation of the result of the auction.

Chapter 3

Auctions

Auctions protocols define the interactions among agents and between agents and the auctioneer during the auction. In this chapter, we present the major categories of auctions and outline their characteristics. A presentation of some protocols proposed for single item auctions and combinatorial auctions follows. Also, an in depth presentation of vMB-Share, which is the protocol that we generalize in this work, is made.

3.1 Auctions Characteristics

One distinguishes four major categories of single item auctions: English auctions, Dutch auctions, First price sealed bid auction and Vickrey auctions.

English auctions: English auction is the most commonly type used in the USA [17]. In an English auction, the seller and the bidders come together to start the auction. The auctioneer presents the resource and fixes a minimum bid value. If a bidder is interested in the item, she proposes a bid greater than the previous bid amount. The bidder whose last bid is the highest wins the auction.

Dutch auctions: In a Dutch auction, the seller and the bidders also come together and start the auction. The seller introduces the resource and sets a very high price. If no bidder reacts to the proposed price, the auctioneer lowers the amount and awaits for some reaction. She continues to lower the price until a bidder accepts to buy the item for the current bid amount.

First price sealed bid auctions: In a first price sealed bid auction, the bidders seal their bids and give it to the auctioneer. No bidder knows the amount bid by another participant. The highest bid wins the auction.

Vickrey auctions: A Vickrey auction is similar to a first price sealed auction. Bids are sealed so that bidders don't know each other bid value. The agent with the highest bid wins the auction; however she does not have to pay the value of her bid but rather the second highest price offered for the item.

A special type of Vickrey auction is the (M+1)-st price auction in which the

winner of the auction pays the M+1 highest price offered for the resource.

To enable for auctions over a computer network, protocols that structure the interaction between participants are needed. Some propositions have been made for both single item auctions and combinatorial auctions. We review some of them in the following two sections.

3.2 Single Item Auctions Protocols

A certain number of protocols have been devised for single item auctions, be it single unit or multi-units [13, 15, 5, 14, 23]. However, most of these protocols require a third party. vMB-Share, a newly developed technique, differentiates itself by enabling a fully private and secure auction without third party. In the next section, we describe this protocol in more details.

vMB-Share vMB-share was introduced by Brandt in 2002. It allows a set of agents and a seller to conduct an auction without involving any third party. Some terminology used:

Definition 7 (Bid vector:) *A bid vector is a tuple where each element corresponds to a certain price. The elements are ordered in the increasing value of the price. These elements belong to a set* Φ *where the cardinality of* Φ *equals 2.* $\Phi = \{0, \lambda\}$ *where* λ *is a publicly known by all the participants and* $\lambda \neq 0$. *The*

number of occurrence of λ equals the value of the bid and the size of the vector equals the number of possible bids. E.g. $\lambda = 1$ and $\Gamma = \{1, 1, 1, 0, 0\}$ (bid value = 3).

Definition 8 (Differential bid vector:) *A differential bid vector is a vector whose elements values also belong to the set Φ where the cardinality of Φ equals 2. One element of the set Φ is 0. $\Phi = \{0, \lambda\}$ where λ is a publicly known by all the participants and $\lambda \neq 0$. The position of the element λ determines the value of the bid and the size of the vector equals the number of possible bids. E.g. $\lambda = 1$ and $\Gamma = \{0, 0, 1, 0, 0\}$ (bid value = 3).*

vMB-Share presents many interesting characteristics:

- The protocol does not require a third party as judge. This increases the privacy and the correctness of the process. No one can be sure that a judge does not cooperate with a subset of bidders.
- All bidders and seller perform the computations simultaneously.
- Only the winner and the seller learn the winning price.
- The protocol is applicable to $M + 1$-st price type auctions.

Algorithm description: The protocol considers a finite number K of possible bidding prices. y is a value commonly known by all participants, e.g. y=1.

- Each bidder i selects a bid $p_i \epsilon \{0, 1, 2, ..., K\}$

- Each bidder i creates a differential bid vector Δb_i and shares it with all other participants. The differential bid vector is such that

$$b_{ij} = \begin{cases} y & \text{if } j = p_i \\ 0 & \text{otherwise} \end{cases}$$

  That is $\Delta b_i = (\underbrace{0, ..., 0}_{i-1}, y, 0, ..., 0)$

- The differential bid vector is "integrated" following the technique proposed in [1]. This outputs an integrated vector obtained by successively summing adjacent pairs of elements of the differential bid vector from position K to position 1. The obtained vector, called bid vector, is of the form: $b'_i = (\underbrace{y, ..., y, y}_{b_i}, 0, ..., 0)$.(1)

- The differential bid vector is also integrated in the reverse way. This is done to mask the bid vector obtained after computations so that only the winner and the seller get to know the winning price. (2)

- Finally, the values of the bid vector obtained in (1) are shifted backward by one position. This gives the protocol the ability to deal with the $(M + 1) - st$ price. (3)

- The vectors obtained from (1), (2) and (3) are multiplied with random non-null coefficients and summed together.

- Only the seller and the winner learn the winning bid.

VMB-SHARE Numerical Sample: In the following, we detail a simplified numerical sample of vMB-Share.

The seller S is auctionning a single item $Item1$; There are 2 bidders (n = 2) and 5 possible prices ($K = 5$); The computations are made in the set of integers $\mathbb{Z}$; $j, b_i \in [1, 5]$; y = 1; The auction is a (M+1)st price type with M = 1. The vector identity is $I = (\underbrace{1, 1, ..., 1}_{K=5})$.

Bidder #1, B_1, secretly selects her bid value \$3; bidder #2, B_2, secretly selects her bid value \$4;

1. Each bidder i chooses random multipliers and creates 2j random polynomial. The constant component of j of these polynomials are made of elements of the bidder's bid vector Δb_i. The remaining j polynomials are used for verification purposes (Section 2.3.4). B_1: Δb_1 = (0, 0, 1, 0, 0)

   B_2: Δb_2 = (0, 0, 0, 1, 0)

2. The bid vector are integrated from component k to 1: b_1 = (1, 1, 1, 0, 0)

   b_2 = (1, 1, 1, 1, 0)

3. Integration in reverse order: b'_1 = (0, 0, 1, 1, 1) b'_2 = (0, 0, 0, 1, 1)

4. The bids vectors from 2. are shifted down by one position: $b"_1$ = (1, 1, 0, 0, 0) $b"_2$ = (1, 1, 1, 0, 0)

5. $b = b_1 + b_2$ = (2, 2, 2, 1, 0)

   b" = b"$_1$ + b"$_2$ = (2, 2, 1, 0, 0)

6. $b + b$" = (4, 4, 3, 1, 0);

   (2M+1)*I. = (3, 3, 3, 3, 3)

   T = $b + b$" - (2M+1)*I = (1, 1, 0, -2, -3)

7. Result vectors: V_1 = T + (2M+2)*b'_1 = (1, 1, 0, -2, -3) +4*(0, 0, 1, 1, 1) = (1, 1, 4, 2, 1)

   V_2 = T + (2M+2)*b'_2 = (1, 1, 0, -2, -3) +4*(0, 0, 0, 1, 1) = (1, 1, 0, 2, 1)

Bidder #2 wins the auction because one element of her bid vector $V_2[3]$ equals 0. She has to pay the corresponding price p_3.

3.3 Combinatorial Auctions Protocols

We present here some existing protocols introduced for solving combinatorial auctions.

Generalized Vickrey Auction (GVA) This algorithm [27] generalizes the Vickrey auction protocol to handle combinatorial auctions. It uses the Clarke-Grove [6, 12] mechanism to determine the amount to be paid by each winning bidder.

Secure Generalized Vickrey Auction using Homomorphic Encryption [26] This is an enhancement of GVA using homomorphic encryption. Bidders' bid values are kept secret and unknown to other agents. A set of trusted servers are required for computing the result of the auction.

Secure Combinatorial Auction by Dynamic Programming with Polynomial Secret Sharing [25] This protocol uses dynamic programming and secret sharing technique to compute the winning bid. A third party is needed to compute the result of the auction. No other information on the bidders bid values is revealed in this protocol.

Several other techniques applicable to combinatorial auctions are described in [22].

3.4 Conclusion

In this chapter, we have reviewed the types of auctions and their characteristics. English auctions, Dutch auctions, First-price sealed bid auctions and Vickrey auctions are presented. $(M+1)-st$ price auctions, a special type of Vickrey auctions, are also introduced. Then we have presented some protocols proposed for single item auctions and combinatorial auctions with an emphasis on vMB-Share.

Chapter 4

Secure Negotiations Solver (SNS)

This chapter presents the algorithm developed within the framework of this master thesis. As stated before, this work addresses a general class of problems that can be solved using combinatorial auctions. The proposed technique allows for clearing markets with multiple buyers and sellers, and offers an important degree of privacy. Other methods have been proposed to solve combinatorial auctions and markets (Chapter 3). This method is based on vMB-share and is a generalization of that technique to combinatorial markets.

Several situations cannot be approached with single item auctions (e.g. an auctioneer offering sets of items that cannot be sold alone or participants in parallel auction processes that are willing to acquire either an entire set of

items or none of them). The concept of auction combination table proposed here allows for using vMB-share by adding an abstraction; namely every candidate allocation is represented by a virtual participant shared among real ones.

In the next sections of this chapter, we present an overview of SNS and review the fundamental notions developed in order to ensure a private, correct and secure auction.

4.1 Secure Negotiations Solver Overview

This section presents the *Secure Nogotiations Solver (SNS)*. As in vMB-share, the agents cooperate using multiparty computation techniques to avoid the need of a trusted party.

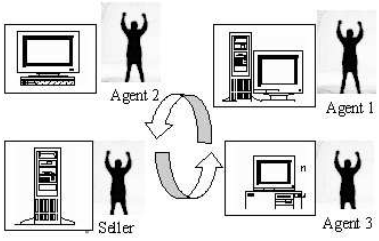


Figure 4.1: Auction process

The algorithm consists of securely distributing shares of the bid for each candidate allocation using Shamir's scheme. A candidate allocation is represented by a tuple t_i specifying the agents to whom each item is attributed.

Tuple $t_i = (A_{i_1}, A_{i_2}, ..., A_{i_n})$

where $t_i[k] = A_{i_k} \Longrightarrow agent\ A_{i_k}\ gets\ product\ k\ for\ allocation\ t_i$.

The shared values are then summed for each tuple and the obtained sum is securely transformed into a differential bid vector using a technique we describe later. Then the problem is transformed to its virtual form: each candidate clearing alternative becomes a virtual agent when applying vMB-share for determining the (M+1)-st highest value from the summed bids.

The following subsections clearly explain the construction of auction combination tables and detail how the differential bid vectors are generated. Then the modifications needed in vMB-share are also described.

4.2 Auction combination table

As previous defined, a single item auction is a process whereby the seller possesses a unique item or a multi-unit item that he is willing to sell by auctioning it off to the agents engaged in the auction. Protocols proposed for single item auction are not suitable for cases where a seller is offering distinct items. For those cases, we introduce the notion of *auction combination table* which is a table of all possible allocation of resources to bidders. Each agent inputs a bid value for every tuple in the table.

Lemma: The number of tuples of an auction combination table when p resources and n agents are involved in the auction is n^p.

Proof: There are n possible distributions of a product and there are p products. The allocations being independent, the total number of resources allocations R, is equal to the product of the number of possible allocation of individual products. Therefore, we have:

$$R = N_{allocation}(item_1) * N_{allocation}(item_2) * ... * N_{allocation}(item_p)$$

$$R = \underbrace{n * n * ... * n}_{p\ times} = n^p$$

Example 1: Suppose we have 1 item to negotiate among four participants (Figure 4.1). The following cases may appear:

1. No agent win the item being sold (this case is part of the domain of possibilities because we allow bidders to bid on the combination where no agent wins the auction. The agent who bids on that combination still has to pay the seller for her bid);
2. Agent A_1 wins the auction;
3. Agent A_2 wins the auction;
4. Agent A_3 wins the auction;

We have a total number of possibilities of 4 and $n^p = 4^1 = 4$.

Combination	Item 1	Item 2	Bid
0	Agent 0	Agent 0	b0
1	Agent 0	Agent 1	b1
2	Agent 0	Agent 2	b2
3	Agent 0	Agent 3	b3
4	Agent 1	Agent 0	b4
5	Agent 1	Agent 1	b5
6	Agent 1	Agent 2	b6
7	Agent 1	Agent 3	b7
8	Agent 2	Agent 0	b8
9	Agent 2	Agent 1	b9
10	Agent 2	Agent 2	b10
11	Agent 2	Agent 3	b11
12	Agent 3	Agent 0	b12
13	Agent 3	Agent 1	b13
14	Agent 3	Agent 2	b14
15	Agent 3	Agent 3	b15

Table 4.1: Sample Agent Table

Example 2: Table 4.1 shows an agent's combination table for that auction when an additionnal item is added to the set of resources. A single seller A_0

proposes 2 items to 3 agents.

We call the representation of negotiations with auction combination table a *virtual form*.

Agent A_0 in the above figure symbolizes that the item remains to the seller. As it can be seen, SNS allows for bidders to bid not only on individual items but also on combination of items. Each bidder may bid on more than one combination of items. If agent A_1 is willing to pay for any item only in the event that she gets both of them, she can bid null values for all the other combinations except the combination #4. If agent A_1 is only interested in item #1 but is not willing to pay for it unless item #2 does not get attributed to her opponent, agent A_2, she can do so by bidding on the combination #3 and 4. If agent A_1 is mainly interested in getting item #2 independently of who gets item #1, the combinations #1, 4, and 7 are suitable for her.

As it can be seen, the protocol is highly flexible and allows for all combinations of items agents might be willing to pay for. Apparently, this flexibility can be considered as a drawback when we consider that agent A_1 can also bid on combination #8 where she does not appear at all. But actually, this does not influence the correctness of the algorithm because SNS first computes the sum of bids offered by all agents for all candidate allocation.

Another approach in building the auction combination table consists of allowing the agents to bid solely on the allocations in which they appear. This

approach is illustrated in table 4.2. This table is obtained by removing from the agent's table 4.1 all combinations that do not allocate a resource to the agent. This method of construction is suitable to cases where the emphasis is put on the bidder herself. Both approaches are compatible with SNS. It is possible to verify that the value of a bid is positive and in a certain range.

Combination	Item 1	Item 2	Bid
0	Agent 0	Agent 0	b0
1	Agent 0	Agent 1	b1
2	Agent 1	Agent 0	b2
3	Agent 1	Agent 1	b3
4	Agent 1	Agent 2	b4
5	Agent 1	Agent 3	b5
6	Agent 2	Agent 1	b6
7	Agent 3	Agent 1	b7

Table 4.2: Sample Agent Table - Type II

Example 3: An example of auction combination table when the protocol is used in a combinatorial market exchange situation is illustrated in table 4.3. We still have four participants. Agent A_0 is selling item 1 and agent A_1 is selling item 2. The reservation price of item 2 is fixed to \$5 by agent 2.

Combination	Item 1	Item 2	Bid
0	Agent 0	Agent 0	-5
1	Agent 0	Agent 1	-5
2	Agent 0	Agent 2	0
3	Agent 0	Agent 3	-5
4	Agent 1	Agent 0	-5
5	Agent 1	Agent 1	-5
6	Agent 1	Agent 2	0
7	Agent 1	Agent 3	-5
8	Agent 2	Agent 0	b8
9	Agent 2	Agent 1	b9
10	Agent 2	Agent 2	b10
11	Agent 2	Agent 3	b11
12	Agent 3	Agent 0	-5
13	Agent 3	Agent 1	-5
14	Agent 3	Agent 2	0
15	Agent 3	Agent 3	-5

Table 4.3: Seller #2 Auction Table

In this table the values for the bids are related to the reserved price fixed at \$5. Agent A_2 may be willing to propose a certain price to acquire item #1. This

is the reason why we input unknown values b8, b9, b10 and b11 for combinations # 8, 9, 10 and 11 respectively. However, agent A_2 is willing to sell her resource (item 2). Therefore, she will not pay for the item to be reassigned to her. This explains the bid value 0 in combinations # 2, 6, and 14. We cannot input 0 at conbination # 10 because here a second item gets assigned to agent A_2 and she might be willing to pay for getting both items. The remaining combinations (0, 1, 3, 4, 5, 7, 12, 13, 15) are assigned the bid value $-5. This motivated by the reserved price of $5 proposed by agent A_2. This price represents the price agent A_2 will receive for her item if the market is cleared.

4.3 Building Differential Bid Vectors

After receiving shares of other participants' bids for all entries in the auction combination table, each agent computes the sum of shares per allocation. These sums are then transformed into differential bid vectors to conform to the format used in vMB-share. Shamir's sharing scheme is a homomorphic sharing scheme. Therefore, performing addition on all shares of the secret is equivalent to performing addition on the secrets themselves (section 2.3.7). A technique is available for multiplication too (section 2.3.8). Consequently, we can perform the differential bid vector transformation at each agent side using shares of the summed value. The formula used for the transformation is

$$V_{t_i,j} = \frac{\prod_{k=0,k\neq j}^{K}(ys_{t_i} - k)}{\prod_{k=0,k\neq j}^{K}(j-k)}$$

s_{t_i} in the formula above represents the share of the total price of allocation t_i. $V_{t_i,j}$ is the value at position j of the differential bid vector corresponding to combination i. Each agent A_i knows only his share $s_{t_i}^{A_i}$. Each secret value $s_{t_i} \epsilon \{0, 1, 2, ..., K\}$. After this computation, each allocation tuple is represented by a shared array of size K with element values γ_i such that

$$\gamma_i = \begin{cases} y & \text{if } s_{t_i} = i \\ 0 & \text{if } s_{t_i} \neq i \end{cases}$$

vMB-share computes Shamir's sharing with elements in $\mathbb{Z}$. We could not use integer numbers in our protocol because the coefficients of the polynomial representing the result of multiplication are jointly generated by all agents. Therefore, these coefficients can be rational numbers. The solution we chose to this problem consists in using modular arithmetic as the computational type of our sharing scheme.

Agents jointly run vMB-share_modified, we will name it cMB-Share, to determine the greatest value among all proposed bids. As we said before, here a substitution is made and the allocations are input as virtual agents.

4.4 Random Multipliers

The output from cMB-share -the modified version of vMB-Share- is a vector where the position in the winning combination corresponding to the $(M+1)-st$ price is 0. The values of the other elements of the vector depend on the values of the bids. Agents jointly compute non-null random multipliers that will be used to hide values of their bids. The single element to remain identical is the element with value 0; the multiplication with the random multiplier keeps its value unchanged. Since we use modular arithmetic in our sharing scheme, the technique used in vMB-share to generate random multipliers is not applicable. We therefore propose a new technique of generating random multipliers for hiding secrets. Each agent generates K random non-zero numbers m_{k_i} and sends their corresponding shares to all other agents. These shares are multiplied together by each participant. The resulting sum is the random multiplier that the agent uses to mask his shares. This stage of the computation generates some computations overhead due to the multiplication of secrets.

4.5 Winning Bids Revelation

For each candidate allocation t_i, agent A_i calculates the representation of the combination t_i in base n by computing the representation of i in base n where n is the number of participants. This representation corresponds to the list of

bidders involved in that particular combination. Agent A_i then sends the array of shares corresponding to t_i to each one of those participants. The seller, agent A_0, is sent all shares.

Example: For the auction depicted in table 4.1, we have n = 4 participants. To get the list of participants to whom shares of combination #6 should be sent, we compute 6 base(4) = 12; Agents 1 and 2 are involved in that combination and therefore the shares have to be sent to both agents.

Only the participants involved in the winning combination get to know the winning combination and the price to pay which is the $(M + 1) - st$ highest amount from the summed prices. Currently, our protocol is compatible with first price sealed bid auctions (M=0) (section 3.1).

The above computations provide the total amount the seller will receive from all winners. It does not yet indicate the price individual winning agents will have to pay. Here, we assume that the seller contacts those agents involved in the winning combination and asks for their individual bids. If the sum of the values received does not match the sum of the selling prices, the auction is null. Another approach which is already used in [26] is to use the Clarke-Grove [6] mechanism to determine the price each winner has to pay. Future work includes integrating that technique in our protocol.

4.6 SNS Algorithm Summary

The algorithm consists of securely distributing shares of bid values of all combinations of products to other agents using Shamir's scheme. Each agent sums the shares from other agents for each candidate allocation and the obtained sum is then securely transformed into a differential bid vector.

The main steps of the generalized auction solver protocol are as follows: Each agent A_k creates the combination table and assigns a bidding value for each combination. For each possible combination, each agent computes the shares of his bid b_i using a (t, n) Shamir's threshold scheme and sends it to all other participants. The agent uses a (t, n) threshold where n = number of participants and $t = \lfloor \frac{n+1}{2} \rfloor$. Each agent then computes the value of $c(t)$ for each combination t of the global problem using existing shares. This is achieved by summing up the shares received from other agents for each combination. Each agent applies the technique described is section 4.3 procedure to compute the shares of the bid vector from the value of the summed share. This is performed for each combination. Apply cMB-share on the differential bid vectors obtained. The number of combinations becomes the number of participants in this section. Jointly compute random multipliers to mask the values of the bid vectors obtained. Multiply the random values with the elements of the bid vectors. Each agent sends the values of his bid vectors shares to other participants. For

each combination, he sends shares only to those participants involved in that combination and to the seller. The seller receives all shares and is able to determine the winning combination and the corresponding price. Each participant in the winning combination gets to know corresponding price. If an element of the differential bid vector for a combination t_i is 0 ($v_{t_i,k}^{A_i} = 0$), the combination t_i is the winning combination and agent A_i is part of that combination. Together with the other agents involved in the combination t_i, she has to pay the price p_k. Figure 4.2 details the step by step algorithm.

Create combination table;
Select bids and send shares;
Sum shares and create differential bid vector;
Run vMB-share;
Multiply with random numbers;
Send shares to other agents;
Determine winning combination;

Algorithm 1: SNS Pseudo-code

1. Create auction combination table. Each tuple t_i of this table has a maximum price of K; $\forall t_i$, A_i chooses a bid $b_{t_i} \in \{0, 1, ..., k_i\}$ and computes n shares, $v_{i,j}^{t_i}$ using a $(\left\lfloor \frac{n+1}{2} \right\rfloor, n)$ threshold scheme.

2. $\forall j$, send agent A_j the share $v_{i,j}^{t_i}$.

3. Computes total bid share for each allocation c_i in table.

$$B_{c_i} = \sum_j v_{j,i}^{t_i}$$

4. Compute differential bid vector

$$V_{t,j} = \frac{\prod_{k=0,k\neq j}^{K} (ys_{t_i} - k)}{\prod_{k=0,k\neq j}^{K} (j-k)}$$

5. Run vMB-share-modified for determining the biggest of all V_{t_i}.

6. $\forall k$, Generate random number m_k and compute shares of it using a (n/2+1, n) threshold.

7. Send $m_{i,j}^{A_j}$ to agent A_j, $\forall j$

8. Compute $m_k = \prod_j m_{i,j}^{A_j}$

9. All agents jointly computes, $\forall t_i$, $v_{t_i,k} = v_{t_i,k} * m_k$

10. Each agent sends the shares $v_{t_i,k}$ to all participants involved in the auction combination table tuple t_i. The agents involved in t_i are known by computing i mod n.

11. The results for each agent are revealed and the winner combination with its value is found (as in other standard protocols).

Figure 4.2: Detailed Algorithm

4.7 Combinatorial Auctions

If the type of negotiation is a combinatorial auction, that is an auction involving a single seller or a group of sellers acting conjointly and multiple bidders and items, the algorithm is identical. However, the determination of individual prices to be received from each winning agent is different. Here, the Clarke-Grove [6] mechanism can be applied to determine a specific amount for each agent involved in the winning combination as described in [26].

4.8 Discussion

vMB-Share makes computations in the set of integers $\mathbb{Z}$. As it was pointed out in section 2.1, our protocol involves division of integers numbers and therefore requires the computations to be made in a field. We chose to make our computations in a Galois field. The order of the field is a randomly generated large prime number.

The determination of the price each winner has to pay can be improved by using Clarke-Groves [6] as implemented in [26]. Another point of improvement is the level of privacy of the algorithm. The current level is $\frac{n}{2}$ which means that if $\frac{n}{2}$ participants collude, they are able to infer some information on the values of secrets. We think that the level of privacy can be increase to n (fully private). This constitutes part of future work.

Chapter 5

Theoretical and Experimental Results

5.1 SNS Protocol Characteristics

In section 1.2, we presented the desired properties of the protocol. In this section, we list each of these properties and for each of them, we specify whether or not the requirement was fulfilled.

- Multiple items: [Fulfilled] Our protocol allows for the seller to propose more than one item to the agents.
- Multiple sellers: [Fulfilled] The protocol allows for more than one seller in a negotiation.

- Seller's constraints: [Fulfilled] The protocol allows a seller to add constraints on his items. He is able to enforce that a subgroup of items be either sold together (not necessarily to the same buyer) or not sold at all.

- Buyer's constraints: [Fulfilled] Each bidder is given the ability to specify a subset of items that she is willing to buy either completely or not at all. This enables her to enforce constraints on items she is willing to pay for. (e.g. if an agent is willing to pay for item #2 only if she has item #1, she will bid for the corresponding combination in the auction combination table. If it is the case that she only wants to buy item #2 and has no interest on the other items, she is also able to do so.

- Privacy: [Partially Fulfilled] The privacy in enforced by dividing the trust onto the bidders themselves. No trusted third-party is involved in the auction. However, we make some multiplication on the values of the secrets. For this reason, the threshold scheme is reduced to $(\lfloor \frac{n+1}{2} \rfloor, n)$ (section 2.3.8). The method offers only $n/2$ privacy: if $\lfloor \frac{n+1}{2} \rfloor$ participants collude, they are able to infer some information on the values of secrets.

- Correctness: [Fulfilled] Our protocol securely input the bids and successfully computes the winning price for each item.

- Bid privacy: [Fulfilled] Only the seller and the winners learn the total winning price of the auction. The price to be paid for each distinct resource

is shared by the seller and the winner of the resource.

- Truth incentiveness: [Not Fulfilled] Our current protocol is compliant with first price sealed bid auctions. However, it can be easily extended to Vickrey auctions and $(M+1)-st$ price auctions.

5.2 Messages Send/Receive

We can compute the expected number of messages exchanged among participants during the auction. This is a measure of the efficiency of the algorithm. We devised a formula that output the number of messages sent/received by each agent. This number of messages is a factor of the number n of agents, the number of items and the total number of prices K. More precisely,

$$Number\ messages = n^{N_{items}} * K + n^{N_{items}} * K * (K-2) + 3$$

- $n^{N_{items}} * K$ represents the number of messages exchanged for determining the winning combination

- $n^{N_{items}} * K * (K-2)$ corresponds to the number of messages exchanged during the transformation of the summed secret to differential bid vector form

- 3 expresses the number of messages sent during the initial sharing phase + computation of commonly shared random multipliers.

$$Number\ messages = n^{N_{items}} * K * (K-1) + 3$$

where is n is the number of participants,

N_{items} is the number of resources presented in the auction and

K is the total number of prices for each allocation.

The component $n^{N_{items}}$ in the formula represents the total number of resources allocation in the auction (cf. section [sec:auctiontable]). From this formula, we have computed the expected number of messages sent for different auction scenarios. Our variables are the number of participants, the number of resources available and the maximum number of prices allowed in the auction. The results are summarized in the following tables and figures.

Number of prices K =	5					
Participants	3	5	10	20	30	40
Items	3	3	3	3	3	3
Auction Combination Table	27	125	1000	8000	27000	64000
Messages	543	2503	20003	160003	540003	1280003

Table 5.1: Messages Exchanged - Items=3

This table shows the expected number of messages exchanged for an auction where 3 distinct resources are proposed to a variable number of participants. Figure 5.1 shows a plot of these data.

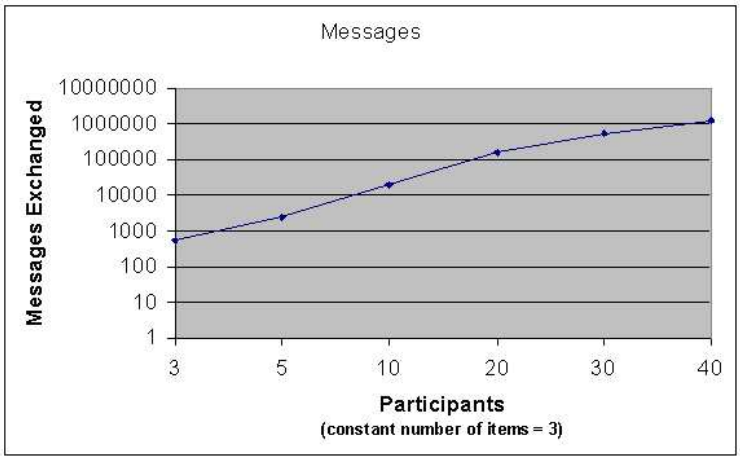


Figure 5.1: Messages exchanged - Items=3

The next table shows the number of messages exchanged by a constant number of participants (5) for different number of available resources. Figure 5.2 shows a plot of these data.

Number of prices K =	5					
Participants	5	5	5	5	5	5
Items	3	5	10	20	30	40
Auction Combination Table Size	125	3125	9765625	9.54E+13	9.31E+20	9.095E+27
Messages	2503	62503	1.95E+08	1.91E+15	1.86E+22	1.819E+29

Table 5.2: Messages Exchanged - Participants=5

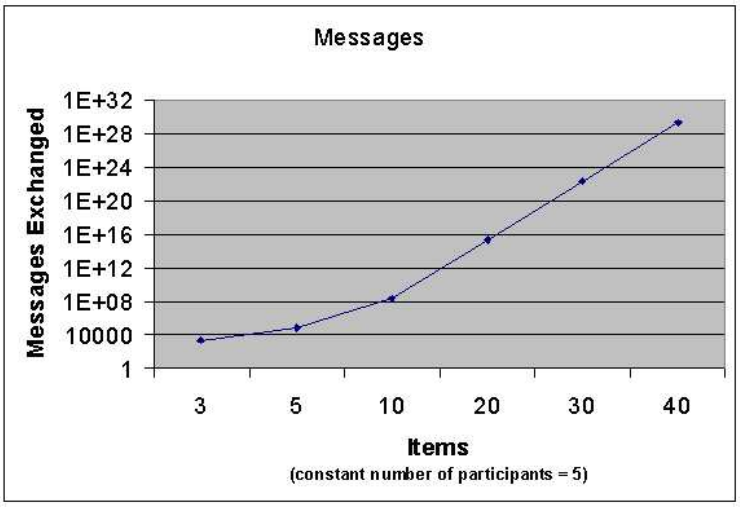


Figure 5.2: Messages Exchanged - Participants=5

A comparison of these plots with those in figure 5.3 and figure 5.4) reveals that the number of messages exchanged depends more on the number of participants and the number of items/resources presented than the number of allowed prices. In these plots, the number of possible prices has been multiplied by a factor of 100 for Figure 5.3. but the increase in the number of messages exchanged is minimal. However, a small increase in the number of participants (from p=5 to p=10) considerably increases the number of messages exchanged (Figure 5.6). Similarly, and even more sensibly, increasing the number of resources presented at the auction greatly influence the computation time (Figure 5.5).

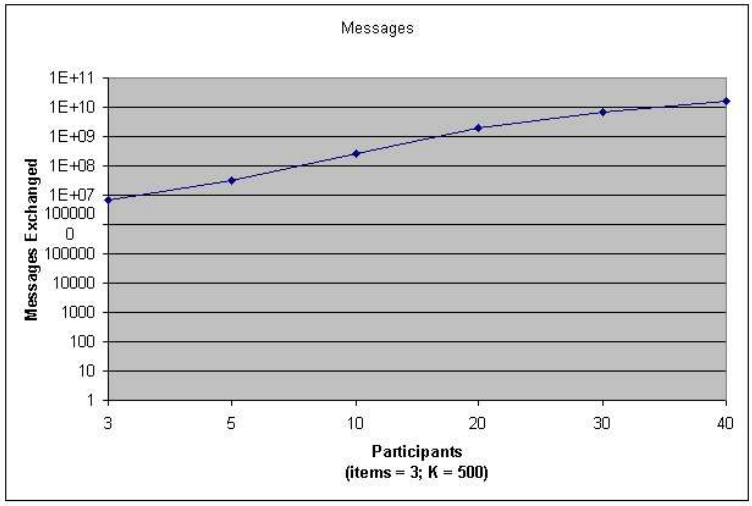


Figure 5.3: Messages exchanged - Items=3;K=500

Number of prices K =	500					
Participants	3	5	10	20	30	40
Items	3	3	3	3	3	3
Auction Combination Table	27	125	1000	8000	27000	64000
Messages	6736503	31187503	2.5E+08	2E+09	6.74E+09	1.6E+10

Table 5.3: Messages Exchanged - Items=3; K=500

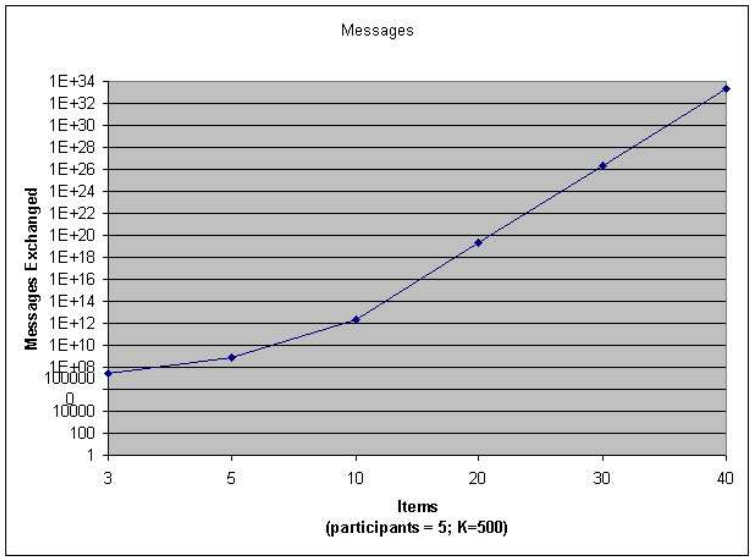


Figure 5.4: Messages exchanged - Participants=5;K=500

Number of prices K =	500					
Participants	5	5	5	5	5	5
Items	3	5	10	20	30	40
Auction Combination Table	125	3125	9765625	9.54E+13	9.31E+20	9.09E+27
Messages	2503	62503	1.95E+08	1.91E+15	1.86E+22	1.81E+29

Table 5.4: Messages Exchanged - Participants=5;K=500

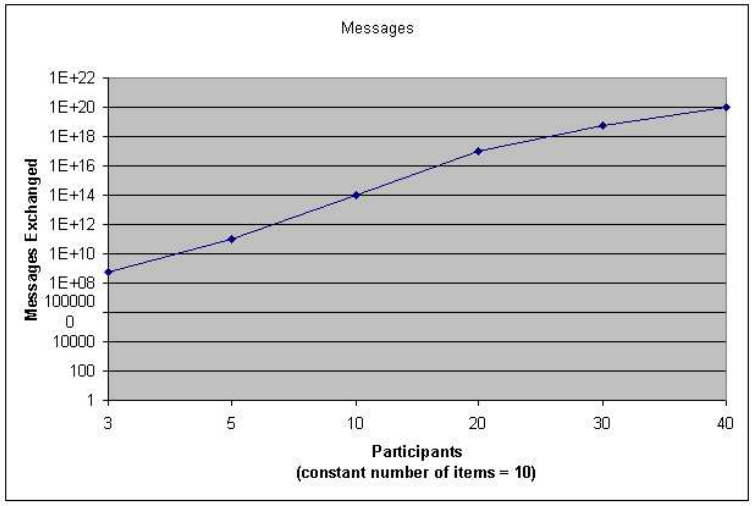


Figure 5.5: Messages exchanged - Items=10;K=100

Number of prices K =	5					
Participants	3	5	10	20	30	40
Items	10	10	10	10	10	10
Auction Combination Table	59049	9765625	1E+10	1.02E+13	5.9E+14	1.05E+16
Messages	5.85E+08	9.67E+10	9.9E+13	1.01E+17	5.85E+18	1.04E+20

Table 5.5: Messages Exchanged - Items=10;K=100

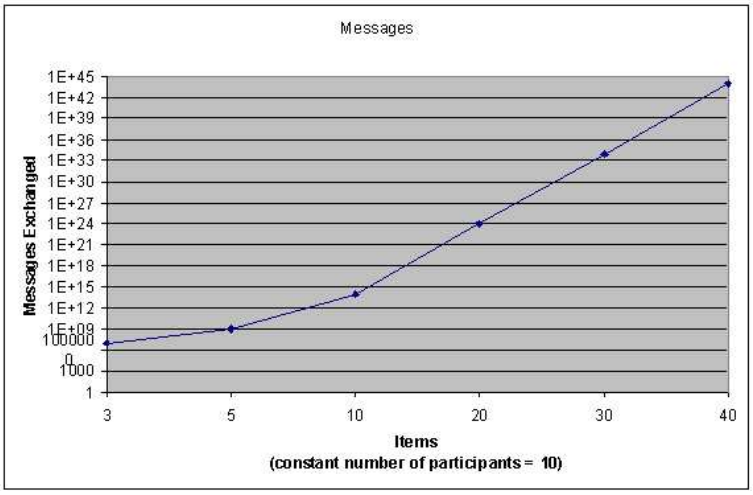


Figure 5.6: Messages exchanged - Participants=10;K=100

Number of prices K =	5					
Participants	10	10	10	10	10	10
Items	3	5	10	20	30	40
Auction Combination Table	1000	100000	1E+10	1E+20	1E+30	1E+40
Messages	9900003	9.9E+08	9.9E+13	9.9E+23	9.9E+33	9.9E+43

Table 5.6: Messages Exchanged - Participants=10;K=100

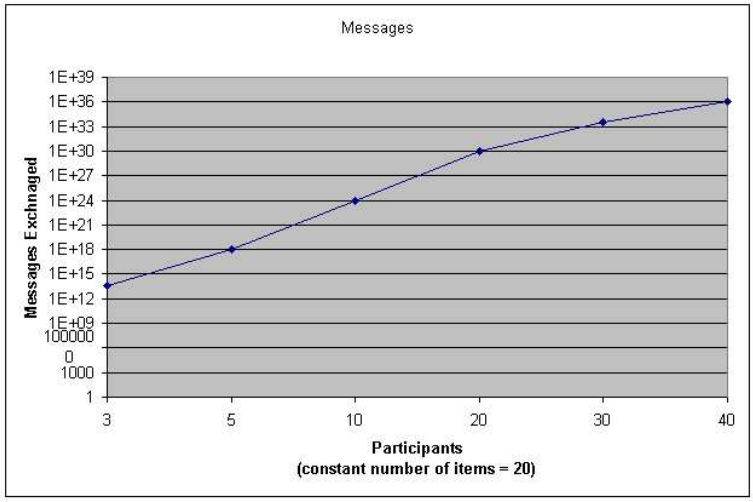


Figure 5.7: Messages exchanged - Items=20;K=100

Number of prices K =	5					
Participants	3	5	10	20	30	40
Items	20	20	20	20	20	20
Auction Combination Table	3.49E+09	9.54E+13	1E+20	1.05E+26	3.49E+29	1.1E+32
Messages	3.45E+13	9.44E+17	9.9E+23	1.04E+30	3.45E+33	1.09E+36

Table 5.7: Messages Exchanged - Items=20; K=100

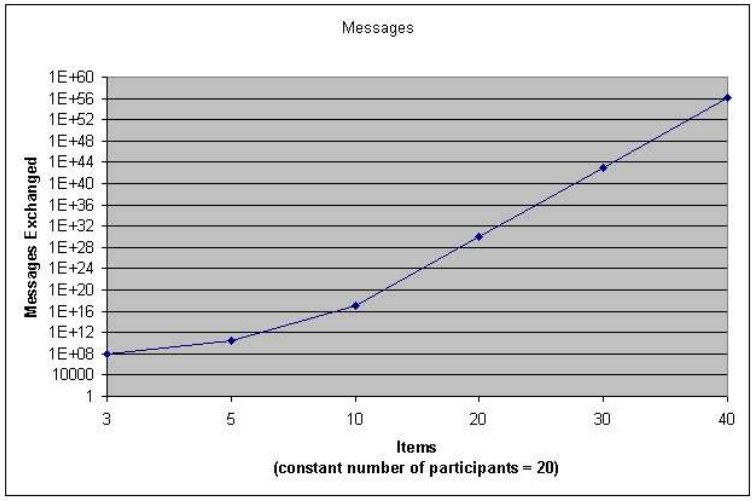


Figure 5.8: Messages exchanged - Participants=20;K=100

Number of prices K =	5					
Participants	20	20	20	20	20	20
Items	3	5	10	20	30	40
Auction Combination Table	8000	3200000	1.02E+13	1.05E+26	1.07E+39	1.1E+52
Messages	79200003	3.17E+10	1.01E+17	1.04E+30	1.06E+43	1.09E+56

Table 5.8: Messages Exchanged - Participants=20;K=100

The number of messages exchanged grows exponentially when the number of items increases. The number of messages exchanged grows logarithmicaly when the number of participants increases. Appendix B shows some numerical data for the case of a constant number of participants of 10, 20 and a constant number of resources of 10, 20 respectively.

5.3 Ties

A tie occurs when more than one participant proposed bid value corresponds to the winning price of the auction. In combinatorial auctions, this can be depicted by more than one candidate allocation sharing an identical total bid value. In this case, the question arises of how to determine the winners among the agents, more than one allocation of resources sharing the highest price.

This situation is not characteristic of combinatorial auctions. In single item auctions, some situations may also result in ties: Two or more bidders could separately propose an equal bid or a group of bidders could purposely decide to input an identical bid value.

We discern two major approaches to the problem:

1. Declare the auction null: This solution is the simplest but also the less secure. A group of bidders who purposely bid an equal amount could learn something about the selling price just by them winning or losing the

process. If the auction fails, they will know that their bid was equal to the selling price. If the tie is not detected, they learn that the amount they proposed was less that the winning price [4].

2. Look for a solution: For single item $(M+1)st$ price auctions, [4] proposed a certain number of methods for handling ties. Two of the suggested methods try to avoid ties whereas the third one attempts to detect the tie value. In overall, these methods increase the computation cost by a factor of $O(n^2kM)$ [4]. We present each of these methods below.

    - Interlacing Vector Component (INT): This method tries to prevent ties from occuring in the first place. The method assigns specific vector positions to each bidder. Ties cannot occur because bids from two bidders cannot be located at the same location. The size of the differential bid vector is increased from K to nK.

    - Preventing Equal Bids (PRE): This technique is similar to the first method. It consists of two phases: detection and correction. The technique first look for ties in bid values. This is done by multiplying the difference of corresponding pair of bids with a random multiplier. For each pair of equal bid found, K rows are inserted in the differential bid vector. This technique is more efficient than the first method: additional rows are not always inserted as in (INT),

but only when needed.

- Determining Ties (DET): This third and last techniqu finds ties (if any) and masks them, allowing the protocol to run normally. Each bidder creates n additional differential bid vectors that are used for masking ties. The protocol is concurrently run on these differential bid vectors with each vector symbolizing a possible number of ties. Only vector i, where i corresponds to the number of ties, outputs a true result to the winner.

SNS makes use of these techniques for dealing with ties. Either method is compatible with SNS and can therefore be used to prevent ties from failing the auction.

5.4 SNS Experimental Results

In table 5.4, we summarize the results of some experimentations of the algorithm. We ran a simulation of the Secure Negotiations Solver algorithm on a SunOs 5.8, 2 Gb RAM machine with different number of participants and different number of items. We kept a constant number of prices $K = 5$. This was done because, as mentioned earlier (section 5.2), the cost of the algorithm is mainly a factor of the number of agents + sellers and the number of resources presented in the auction.

The above experimental results assume that a message being sent will take a single CPU-unit time to arrive at destination. This is due to the fact that all computations are local. Real life situations need to encounter for a message transmission time. The figure 5.9 shows a graph of of the algorithm running time as a function of message transfer duration.

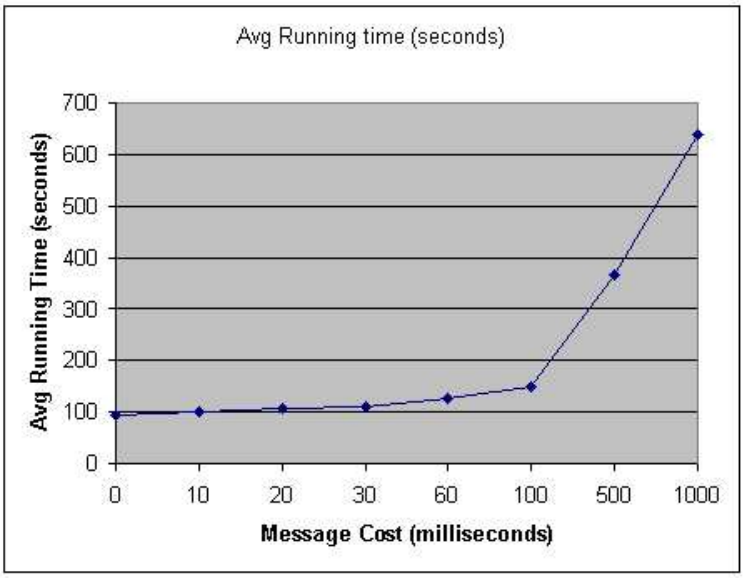


Figure 5.9: Real time Graph

3 participants, 3 items; combinations = 27.	
Time Run:	The auction is completed in an average of 1min33s clock time. *Experiment 1: 1min31s; Experiment 2: 1min34s; Experiment 3: 1min33s*
Messages Exchanged:	We had a total number of messages of 545 sent and received by each participant. Identical in all experiments runs.
3 participants 5 items combinations = 243.	
Time Run:	The auction was completed in around 2min20s. *Experiment 1: 2min24s; Experiment 2: 2min16s; Experiment 3: 2min24s*
Messages Exchanged:	We had a total number of messages of 4865 sent and received by each participant. Identical in all runs of experiments.
5 participants; 5 items; combinations = 3125.	
Time Run:	The auction was completed in around 1hour00 min 09s.
Messages Exchanged:	We had a total number of 62507 messages sent and received by each participant.

Table 5.9: Experimental Results

Chapter 6

Conclusion

The thesis presents a technique for solving combinatorial market exchanges based on an extension of the vMB-Share protocol presented in [3]. vMB-Share [3] is a recent technique, allowing for secure solving of single item auctions, namely where a single seller is offering a single item or multiple unit of a single item to multiple sellers. This algorithm is remarkable due to the high degree of privacy that it offers. No trusted party or auctioneer is needed in any way, as the participants decide the allocation between themselves. Moreover, no subset of participants can find anything except the part of the solution that concerns them. Namely, the winner learns the sum it has to pay, and the auctioneer learns the price it will receive and the identity of the winner to whom it had to give the object of the auction.

We show how to extend vMB-Share to combinatorial auctions. The proposed

technique is applicable to combinatorial market exchanges. Two main elements of the description are:

- Showing how a general negotiation problem can be reduced to what we call the virtual form. This is a specification where each candidate allocation of the negotiated items is treated as a virtual agent, namely a virtual bidder to the clearing of the auction.

- The vMB-share technique is adapted to solve negotiations described in the virtual form. Several modifications were required:

    1. We show that the computations can no longer be done in $\mathbb{N}$. One can use either modular arithmetic or rational numbers. In our experiments we used modular arithmetic for the computation.

    2. The technique used by vMB-share to multiply secrets with random numbers is not applicable with modular arithmetic. We show how an alternative can be constructed. This alternative is related with another algorithm used in [4]. Use of multiplications brings additional overhead, but they are possible as we need (n/2)-privacy schemes for obtaining the virtual form.

    3. A new technique is needed to reveal results only to involved bidders. Such a technique is proposed. It can be improved and some suggestions were made.

To secure its degree of privacy, the running time needs to be problem independent. As our problem is NP-complete, it implies that the running time will always be exponential. Our technique was implemented and some preliminary tests have shown that despite its intrinsic exponential cost, it can be used successfully for some very small real problems.

Appendix A

Attacks on Messages

Encryption of messages prevents unauthorized parties from getting any meaning out of it in case the messenge is intercepted. The sensitivity of a message is the key element that gives the message its value but is also why the opponent will deploy a lot of effort in trying to access it. Attacks that can be made against a message or a system are divided in two categories: passive attacks and active attacks. A passive attack observes the communication or data being sent and attempts to get some information on the message transmitted. An active attack will try to modify or destroy the communication or data being sent.

The principal types of attacks on a message are summarized below [24]:

- Interruption: the message being sent is interrupted on the way and never arrives at the receiver. An example is the physical destruction of a communication line between parties in communication.

- Interception: An unauthorized person gets access to the content of the message. The message still reaches destination. E.g. copying stream of data sent to a particular IP address.

- Modification: The message being sent is modified on the way without either the sender or the receiver having any knowledge of it. The receiver receives a message different from the original on sent by the sender.

- Fabrication: Misrepresentation of a party in communication. An unauthorized party M may contact an agent A and represent itself as a well known partner B of agent A. M will communicate with A and might be revealed confidential information, because agent A assumes the communication is conducted by partner B.

Figure A.1 summarizes the security threats against a message or system.

Cryptosystems must dispose of some security requirements in order to be strong against these threats. The main requirements are confidentiality, authentication, integrity, non-repudiation, access control and availability. These characteristics are important parameter of a cryptosystem. We provide a review of each of them below [24].

- Confidentiality: The cryptosystem should ensure the confidentiality of the message. In other words, the message should only be accessible to the

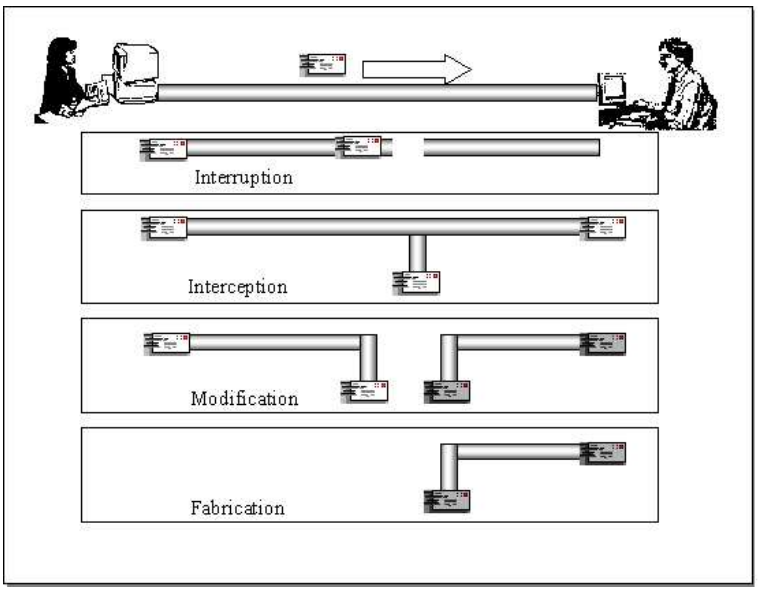


Figure A.1: Attacks on a Message

parties that it is intended for. No unauthorized party should be able to have access to the message. This is a protection against passive attacks.

- Integrity: The receiver should receive the message as sent by the originator. A message maintains is integrity when the cryptosystem insures that there has been no insertion or modification of the content of the original message.

- Authentication: This assures each party in communication that the other party or parties are truly who they claim to be. The recipient is therefore certain that the message really comes from the sender and the sender is assured that the message is truly sent to the recipient. In either way, this protects against any third party pretending to be any one of the parties

for the purpose of getting access to the communication or sending false information.

- Non-repudiation: The cryptosystem should allow the non-repudiation of messages. This assures the parties in communication that the originator of a message cannot deny later having sent it.

- Availability: This ensures that the legitimate users have access to information when they need it.

- Access control: This ensures that unauthorized users are kept out of the resources they should not have access to.

Bibliography

[1] M. Abe and K. Suzuki. M+1-st price auction using homomorphic encryption. In Springer, editor, *5th IC on PKC*, volume 2274 of *LNCS*, pages 115–224, 2002.

[2] G. R. Blakley. Safeguarding crypographic keys. volume 48, pages 313–317, 1979.

[3] Felix Brandt. A verifiable, bidder-resolved auction protocol. In L.Korba R.Falcone, editor, *Deception, Fraud and Trust in Agent Societies (AAMAS-W5)*, pages 18–25, Bologna, July 15 2002.

[4] Felix Brandt. Fully private auctions in a constant number of rounds. In *Financial Cryptography*, Guadeloupe, 2003.

[5] C. Cachin. Efficient private bidding and auctions with an oblivious third party. In *6th ACM Conference on Computer and Communications Security*, pages 120–127, 1999.

[6] E. H. Clarke. Multipart pricing of public goods. volume 2, pages 18,25. Public Choice, 1971.

[7] Federal Communications Commission. http://www.fcc.gov/.

[8] Houghton Mifflin Company. The american heritage dictionary of the english language, fourth ed. 2000.

[9] Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin J. Strauss, and Rebecca N. Wright. Multiparty computation of approximations. volume 2076, pages 927+. Lecture Notes in Computer Science, 2001.

[10] P. Feldman. A Practical Scheme for non-interactive verifiable secret sharing. In *IEEE Fund. of Computer Science*, pages 427–437, 1987.

[11] Michael Frei. http://www.cs.cornell.edu/courses/cs513/2000SP/SecretSharing.html, 2000.

[12] Theodore Groves. Incentives in teams. volume 41, pages 617–631. Econometrica, 1973.

[13] Michael Harkavy, J. D. Tygar, and Hiroaki Kikuchi. Electronic auctions with private bids. In *3rd USENIX Workshop on Electronic Commerce*, pages 61–74, September 1998.

[14] Kikuchi, Hakavy, and Tygar. Multi-round anonymous auction protocols. *TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems*, 1999.

[15] Hiroaki Kikuchi. (m+1)st-price auction protocol.

[16] Inc. MICRA. Webster's revised unabridged dictionary. 1996.

[17] Kate Reynolds. http://www.agorics.com/Library/auctions.html, 1996.

[18] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.

[19] Tuomas Sandholm and Subhash Suri. Market clearability. In *IJCAI*, pages 1145–1151, 2001.

[20] Bruce Schneier. Applied cryptography. John Wiley & Sons, Inc, 1996.

[21] A. Shamir. How to share a secret. In *Communications of the ACM*, pages 22:612–613, 1979.

[22] Marius C. Silaghi. An algorithm applicable to clearing combinatorial exchanges, September 2002.

[23] D.X. Song and J. Millen. Secure auctions in a publish/subscribe system. http://www.csl.sri.com/users/millen.

[24] William Stallings. Cryptography and network security, principles and practices. Prentice Hall, 2002.

[25] Koutarou Suzuki and Makoto Yokoo. Secure combinatorial auction by dynamic programming with polynomial secret sharing. In *6th International Financial Cryptography Conference (FC-02)*, 2002.

[26] Koutarou Suzuki and Makoto Yokoo. Secure generalized vickrey auction using homomorphic encryption. In *Financial Crypography*, pages 239–249, 2003.

[27] Hal R. Varian. Economic mechanism design for computerized agents. In *First Usenix workshop on Electronic Commerce*, 1995.

[28] W. Vickrey. Counter speculation, auctions, and competitive sealed tenders. In *Journal of Finance*, pages 16(1):8–37, 1961.

[29] Andrew C. Yao. Protocols for secure computations. pages 160–164. 23rd Annual IEEE Symposium on Foundations of Computer Science, November 1982.