Secret Sharing In Sealed-Bid Auctions

Maciej Medyk

Video Available On YouTube At The Link : https://www.youtube.com/watch?v=G5q5JwBPyFI

Agenda

- What is sealed-bid auction?
- Where is sealed-bid action used?
- What is secret sharing?
- Review of common secret sharing schemes used in auctions
- Demonstration
- Conclusion

What Is Sealed-bid Auction?

- It's a process in which all bidders submit sealed (encrypted) bids to the auctioneer
- No single bidder knows how much other bidders have bid
- Bidder can only submit one bid per auction
- Bidding normally happens simultaneously
- There are two types of sealed-bid auctions
  - First-price - FPSB
  - Second-price – SPSB – Vickrey

Where Is Sealed-bid Action Used?

- Sealed-bid are used in obtaining government contracts
- Allocating natural resources like oil or timber rights
- Art and artifacts auctions
- Real-estate transactions
- Mobile commerce
- Electronic voting

Auction Properties

- Confidentiality – bid is confidential to anyone other than bidder himself
- Privacy – privacy of losing bids must be retained after auction closes
- Unchangeability – once bids are submitted they cannot be modified
- Verifiability – participants must be able to verify auction outcome
- Fairness – no one can modify or reject submitted bids
- Correctness - determine a winner and announce winning bid
- Robustness – auction can run properly in abnormal, malicious situations

What Is Secret Sharing Protocol?

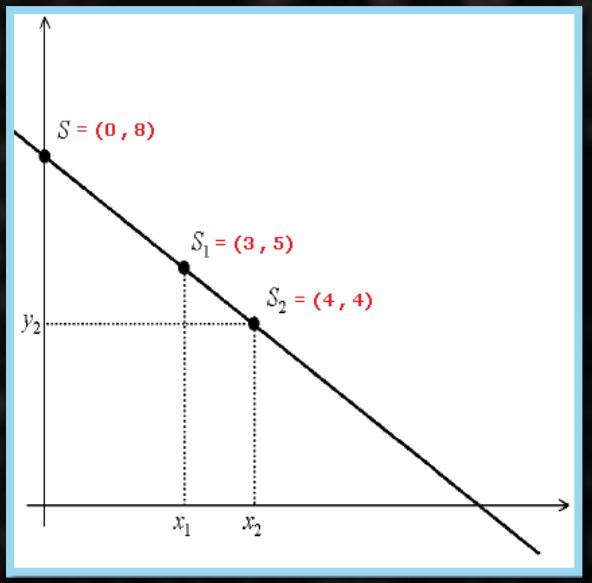
The most simple secret sharing protocol has two phases

- Method of distributing secret amongst group of participants
- Method of reconstructing secret from all individual shares

The most basic example of secret sharing scheme is trivial secret sharing using bit exclusive mathematics

First polynomial secret sharing algorithm has been developed in 1979 by Adi Shamir. It creates shares using polynomial points in order to hide secret f(0)

Secret													
25135 converts to	=	1	1	0	0	1	0	0	0	1	1	1	1
Shares distributed													
Share 1		1	0	1	0	1	0	1	0	1	1	0	0
Share 2		0	0	0	0	0	0	1	1	0	0	1	1
Share 3		1	1	1	1	1	0	0	0	0	0	1	1
Share 4		1	0	0	1	1	0	0	1	0	0	1	1

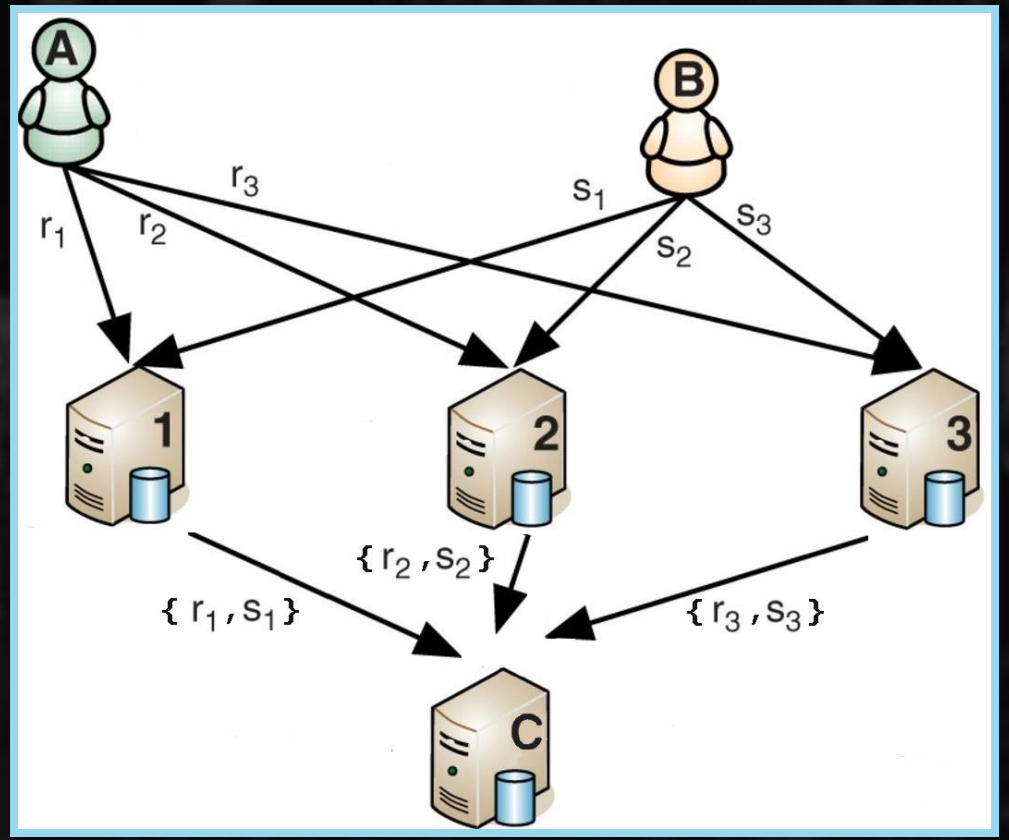


Common Security Schemes Used In Sealed-bid Auctions

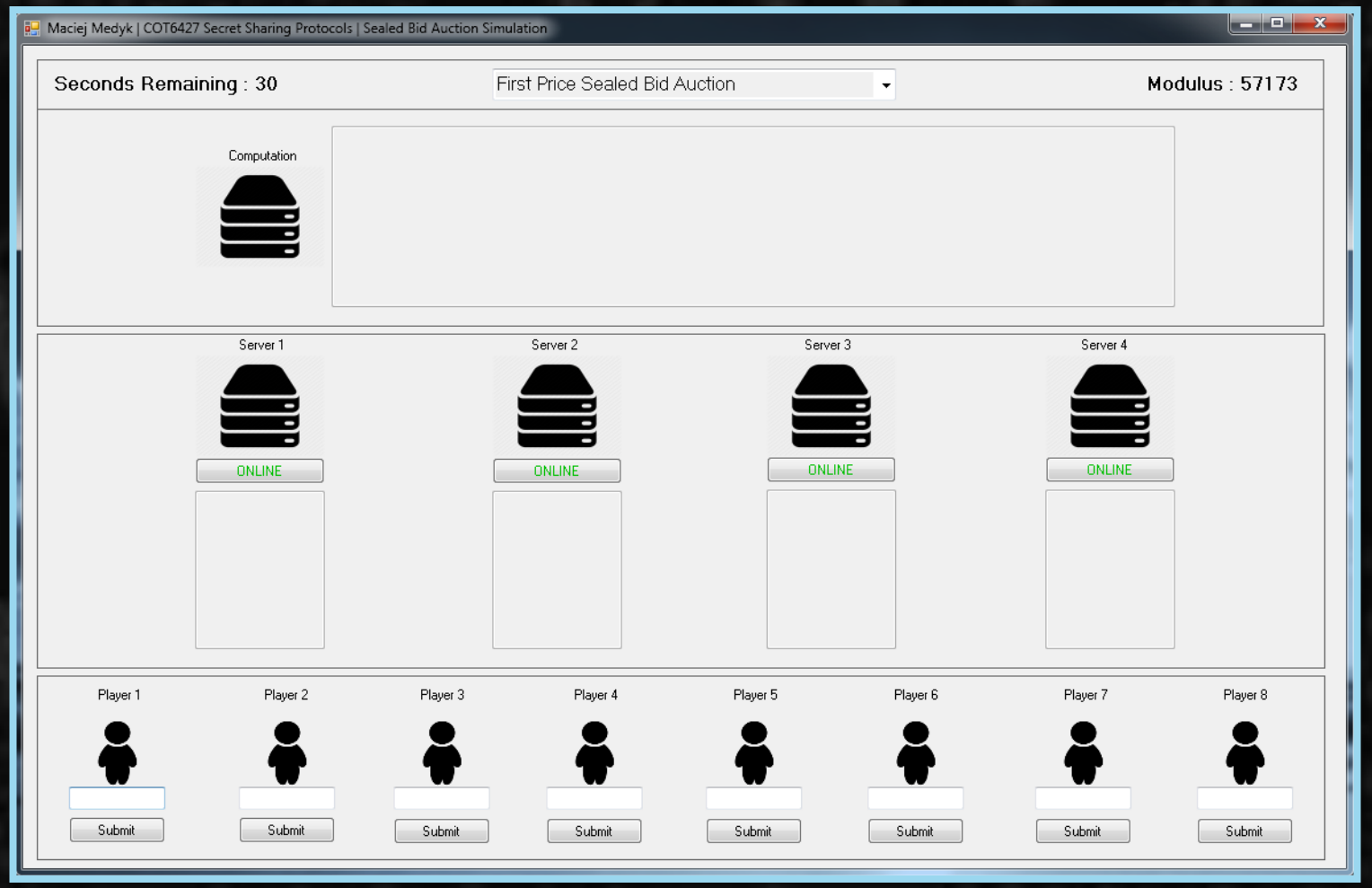
- Shamir's Secret Sharing
- Verifiable Secret Sharing
- Homomorphic Secret Sharing
- Secure Multiparty Computation
- Knapsack

Shamir's Secret Sharing

- Bidders construct shares using polynomial and send the shares to auctioneers
- Shares are being collected by each server until the auction ends
- At the end of the auction the shares are being used to recreate bid using Lagrange interpolation method
- Layer of auctioneer servers provides anonymity to all bidders and threshold of polynomial allows for some auctioneer servers to go offline

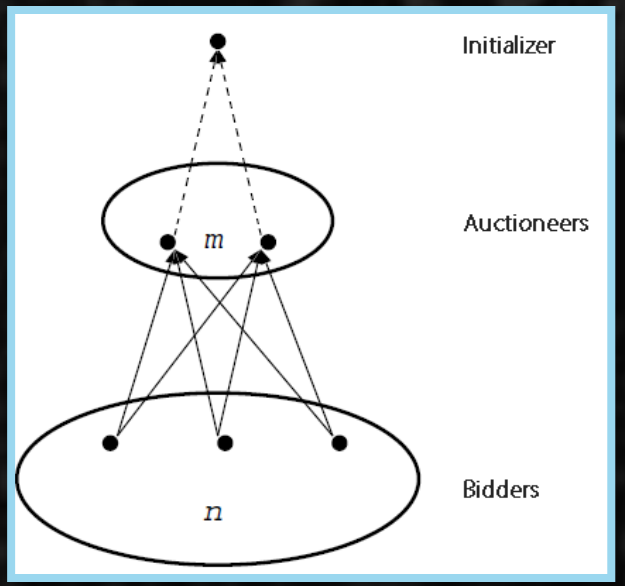


Demonstration



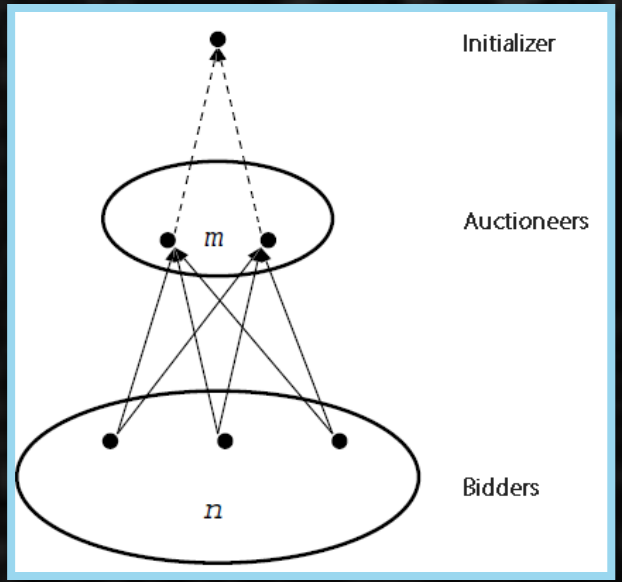
Verifiable Secret Sharing

- Uses bivariate polynomial function where f (0,0) = secret
- No trusted party necessary as bids can be resolved by auctioneers
- Auctioneers receive bids and perform arithmetic operation to hide bids
- At the end of the auction the masked bids are reviled showing differences between bids not values
- Once auction ends, verification of auctioneers is done by cross evaluation to declare a set of good auctioneers.



Verifiable Secret Sharing

- Once the shares are computed they are sorted in descending order and winner is determined
- Auctioneers send winners index and shares to all the bidders through the private channels together with price
- The bidders recreate locally their shares they received and compare them to their bid
- Winning bid is verified by local reconstruction by the bidders
- At the end of protocol bidders only know auction outcome and losing bids are kept secret



Verifiable Secret Sharing

Security Issues

- Auctioneers can collude amongst each other to retrieve secret
- The dishonest shares can be sent to disrupt auction

Security Resolution

- Threshold of the polynomial must be higher than expected amount of dishonest auctioneers
- Use of error correction technique like pairwise checks or Reed-Solomon codes during interpolation to minimize impact of dishonest shares

Homomorphic Secret Sharing

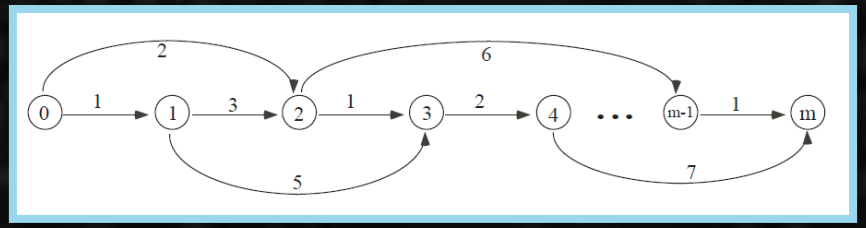
- Homomorphic is type of encryption that allows arithmetic operation on ciphertext without decryption
- Encryption uses random number making encryption indistinguishable Encryption of price 5 will differ depending on random number
  - Using random number 5 → $(5^5 \bmod 23, 2^5 \times 5 \bmod 23) = (20,22)$
  - Using random number 6 → $(5^6 \bmod 23, 2^6 \times 5 \bmod 23) = (8,21)$
- Binary search is used to determine winning bid while encrypted
- Bid validity check is used to assess the correctness and fairness of the homomorphic auction

Homomorphic Secret Sharing

- Uses partially homomorphic encryption developed by El Gamal, Paillier, or Goldwasser-Micali
- The bid is randomized and encrypted as vector of ciphertext

$$(b_{i,1}, b_{i,2}, \ldots, b_{i,w})$$

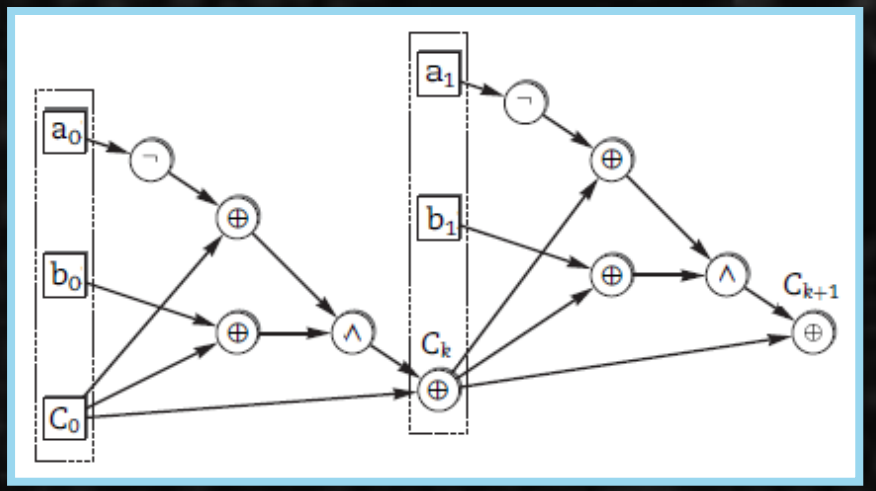
- Constant is added that bid price is further hidden after decryption
- Finding of maximum encrypted price is done by comparing components of vector either using binary search or by using dynamic programming



Secret Multiparty Computation

- Uses comparison function that is build on Boolean circuits
- In comparison each number is presented as vector that is compared with vector of another number
- Once comparison is done the bids are sorted and winner bid is announced
- The shares submitted must be using a threshold secret sharing protocol

$$a_{n-1}2^{n-1} + \ldots + a_1 2^1 + a_0 2^0$$
$$b_{n-1}2^{n-1} + \ldots + b_1 2^1 + b_0 2^0$$
$$A = B + (2^n - 1 - A) + 1 = B + \overline{A} + 1$$
$$C_{k+1} = ((B_k \oplus C_k) \wedge (\neg A_k \oplus C_k)) \oplus C_k$$



Secret Multiparty Computation

- Commitment Transfer Protocol (CTP) allows for transfer of secret from one party to another
- Commitment Sharing Protocol (CSP) allows to share secret in verifiable way
- Commitment Multiplication Protocol (CMP) allows to prove that committed secret shares
- Supports active, static adversary model

Knapsack

- Bidder shares secret using additive shares to hide their bids
- Each bidder divides his secret into random shares and sends their shares to every other bidder to so they can compute collected shares

Initiation

prices	10	100	200	250	300	350	400	450	500
indexes	5	9	15	30	60	120	250	500	1000

modulus	1987	>>	900	700	300	87

Bid Creation

	actual		index		mod		
bidder 1	10	=	5	+	900	mod 1987	905
bidder 2	200	=	15	+	700	mod 1987	715
bidder 3	450	=	500	+	300	mod 1987	800
bidder 4	350	=	120	+	87	mod 1987	207

Share Creation

bidder 1 splits 905	=	200 + 200 + 200 + 305
bidder 2 splits 715	=	100 + 115 + 400 + 100
bidder 3 splits 800	=	150 + 250 + 100 + 300
bidder 4 splits 207	=	50 + 70 + 80 + 7

bidder 1 adds 500	=	200 + 100 + 150 + 50
bidder 2 adds 635	=	200 + 115 + 250 + 70
bidder 3 adds 780	=	200 + 400 + 100 + 80
bidder 4 adds 712	=	305 + 100 + 300 + 7

$$q8 = 500 + 635 + 780 + 712 = 2627 \bmod 1987 = 640$$

Knapsack

- Winner determination is computed using knapsack problem

is (640 >= 1000)	>>	NO	>>	0	>> 640
is (640 >= 500)	>>	YES	>>	1	>> 640 - 500 = 140
is (140 >= 250)	>>	NO	>>	0	>> 140
is (140 >= 120)	>>	YES	>>	1	>> 140 - 120 = 20
is (20 >= 60)	>>	NO	>>	0	>> 20
is (20 >= 30)	>>	NO	>>	0	>> 20
is (20 >= 15)	>>	YES	>>	1	>> 20 - 15 = 5
is (5 >= 9)	>>	NO	>>	0	>> 5 - 5 = 0
is (5 >= 5)	>>	YES	>>	1	>> 5 - 5 = 0

final set of indexes = {1, 0, 1, 0, 0, 1, 0, 1, 0}

prices = {10, null, 200, null, null, 350, null, 450, null}

prices	10	100	200	250	300	350	400	450	500
indexes	5	9	15	30	60	120	250	500	1000

modulus	1987	>>	900	700	300	87

	actual		index		mod		
bidder 1	10	=	5	+	900	mod 1987	905
bidder 2	200	=	15	+	700	mod 1987	715
bidder 3	450	=	500	+	300	mod 1987	800
bidder 4	350	=	120	+	87	mod 1987	207

$$q8 = 500 + 635 + 780 + 712 = 2627 \bmod 1987 = 640$$

Conclusion

- Threshold secret sharing protocols do not require all shares to retrieve the secret as long as threshold is less than number of auctioneers making the protocol very roubust
- Verifiable secret sharing can work without trusted third party
- Secure multi-party often is paired with threshold sharing algorithm like VSS to create shares
- Fully homomorphic applications not practical and partially homomorphic applications are secure
- Knapsack is not adaptable to malicious behavior

Conclusion

Thank You