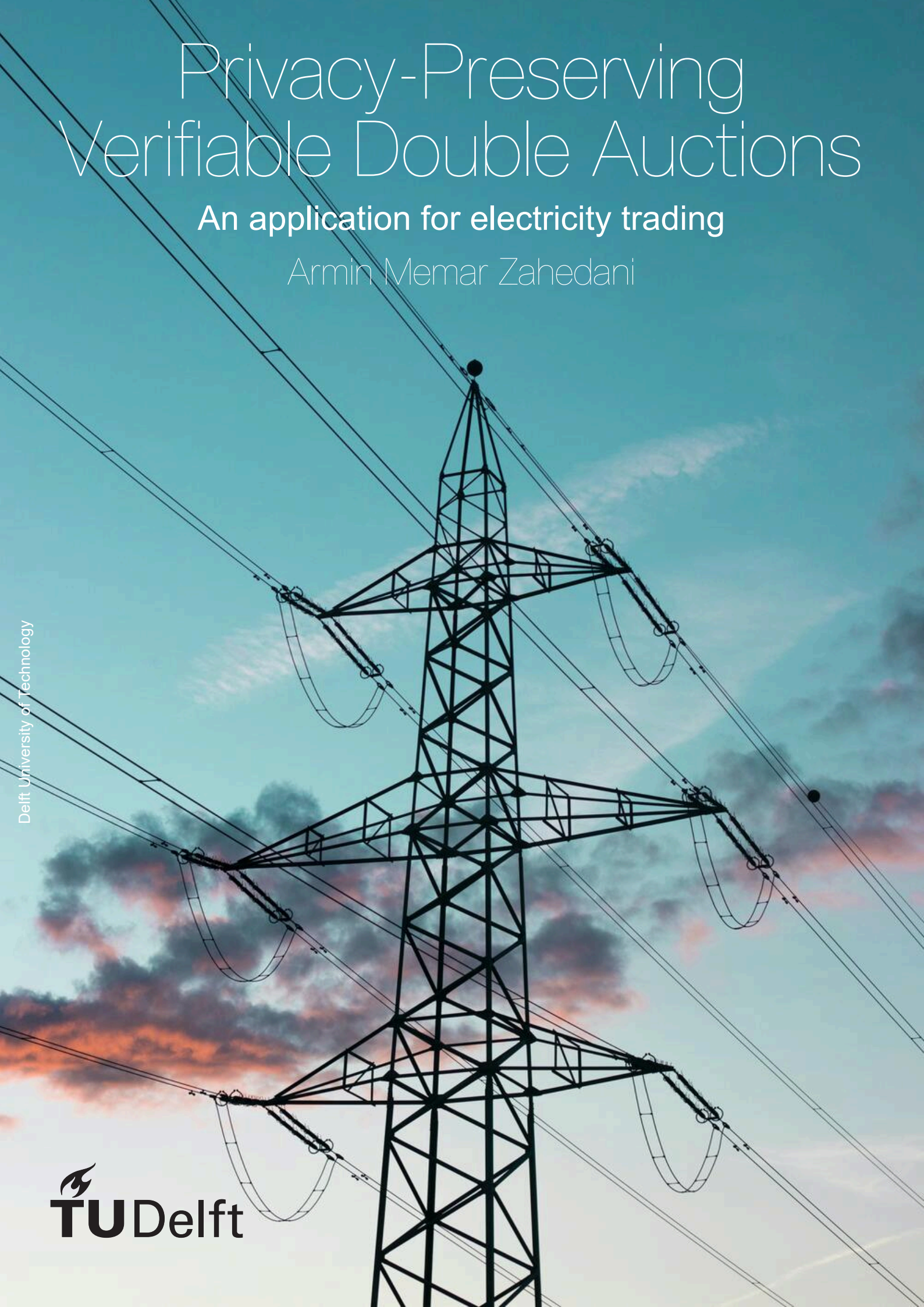
Privacy-Preserving Verifiable Double Auctions

An application for electricity trading

Armin Memar Zahedani



Delft University of Technology

TU Delft

Privacy-Preserving Verifiable Double Auctions

An application for electricity trading

by

Armin Memar Zahedani

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Thursday July 7, 2022 at 09:00 AM.

Student number:	5653121	
Project duration:	November 16, 2021 – July 7, 2022	
Thesis committee:	Dr. Z. Erkin	TU Delft, Supervisor
	Prof. M. Conti	TU Delft
	Dr. S. Roos	TU Delft

An electronic version of this thesis is available at `https://repository.tudelft.nl/`.

TU Delft

Preface

Cryptography was always something I found engaging. Being introduced to the famous RSA encryption scheme during my bachelor's, I knew I wanted to venture into the depths of cybersecurity during my master's. There is just something about the elegance of cryptographic protocols and an excitement to learn about them that motivated me. Studying cybersecurity at TU Delft ignited my passion for security even more as I ventured deeper into various topics. Even though my path through security took a few detours, I arrived again at the subject that introduced me to security in the first place: Cryptography.

The idea to work on auctions for my thesis came from a general interest in finance. It turns out that stock trading is just a type of auction, which led me to study the problem of privacy in auctions. While reading a paper on danish sugar beets auctions, I quickly realized that I would like to write my thesis in this field. Hence, this thesis presents a combination of personal interests in stock trading and auctions and a technical interest in cryptography and its applications that I am thrilled to present.

Firstly, I want to thank my supervisor, Dr. Zekeriya Erkin, for this unique opportunity to combine my interests to form my master thesis. Thank you for continuously challenging me, supporting me throughout this thesis, and pushing me to produce the highest quality work. I want to also thank Jelle Vos for being my daily supervisor, discussing ideas with me, and always having an open ear, whether academic or personal. Furthermore, I would like to thank Prof. Mauro Conti and Dr. Stefanie Roos for being part of my thesis committee, reading my work, and taking the time for my upcoming defense.

Special shoutouts go to students and staff on the 6th floor, including cybersecurity and other research groups that made the 6th floor such a pleasant and fun work environment. Whether talking over lunch or coffee breaks, I truly enjoyed the time and jokes, and I will most certainly miss it.

Five years of bachelor's and master's have flown so fast. I want to thank my friends, especially Linus, Luke, Marco, and Sven, for being part of this journey, learning and discussing with each other, and always being there, especially throughout the pandemic. Lastly, I would like to thank my family for their continuous and unconditional support throughout the last months and years.

Armin Memar Zahedani
Rotterdam, June 2022

Abstract

Double Auctions are mechanisms to trade commodities such as electricity or parts of the wireless spectrum at optimal prices. Bidders and sellers simultaneously submit quantity-price pairs to an auctioneer, denoting the quantity they want to buy or sell at specific prices. The auctioneer aggregates the offers into demand and supply curves to compute the auction result by finding the intersection between supply and demand. In this way, commodities exchange owners in an economically efficient manner, driven by the market. In an ideal scenario, the auctioneer is a trusted third party that does not abuse the information they gain. However, in reality, offers reveal usage patterns of customers, such as electricity usage, or may be used by the auctioneer for their economic gain as insider information. The auctioneer also has opportunities to manipulate results of which there are real-life allegations in electricity trading or advertisement auctions. These concerns call for solutions that conduct the auction in a privacy-preserving and verifiable way while not compromising the auction functionality or economic efficiency.

Proposed solutions for privacy-preserving and verifiable double auctions offer confidentiality but do not allow participants to verify results independently or vice versa without interaction of participants in the full auction procedure. We specifically focus on electricity trading to design a solution covering the above concerns. To the best of our knowledge, we propose the first privacy-preserving and verifiable double auction scheme without interactivity of all participants, tailored to electricity trading on (inter)national exchanges. Using cryptographic schemes, including Homomorphic encryption, Commitment schemes, and Zero-knowledge-proofs, we propose a solution to establish a double auction protocol that preserves privacy and allows for verification.

Contents

Nomenclature

Definitions

Abbreviation	Definition
Dark pool	Private exchanges to trade stocks. Used to give buyers and sellers the opportunity to place large volume orders without the public knowing to not affect the price through large buy or sell orders
Economic Rationale	Acting with economic goals in mind. For the auctioneer, this means to not exclude random participants from the auction
Market-clearing-price	The price at which traders trade the most quantity in the market
Order	An order is the message containing several quantity-price pairs and other relevant information that signalizes to buy or sell the item

Abbreviations

Abbreviation	Definition
DDH	Decisional Diffie-Hellman assumption
EC(C)	Elliptic-curve (cryptography)
ECEG	Elliptic-curve-ElGamal
HE	Homomorphic encryption
MCP	Market-clearing price
MPC	Multiparty computation
MWh	Megawatt-hour
REMIT	Regulation on Wholesale Energy Market Integrity and Transparency
ZKP	Zero-knowledge proof

List of Figures

List of Protocols

1 Introduction

Auctions are the de-facto standard to trade items at prices driven by the market. They allow participants to bid on products for the market to find the optimal trading price, which is useful when there is uncertainty about a suitable market price. There are multiple kinds of auctions that people use for trading different items. Items include commodities such as electricity or the wireless spectrum, tulips in the Netherlands, or antiques and paintings. One example is the *English auction*, in which buyers bid higher than the current highest bid until no buyer is willing to increase their bid. These are used to sell, e.g., arts or antiques [1]. However, other procedures are possible such as *Dutch auctions*, which work in reverse. Procedures also vary depending on whether an auction is *open* or *sealed*. In open-bid auctions, a party observes the bids of other parties, while in sealed-bid auctions, they cannot observe these bids, thus protecting the confidentiality of bids. In the correct setting, sealed-bid auctions have desirable economic properties, such as truthfulness. The optimal strategy for bidders is to bid based on the value they assign to items and not on what they believe others will bid. In *double auctions* both buyers and sellers submit offers, and the auction finds an equilibrium to trade.

While presenting an efficient, economical process to compute a market price, auctions rely on a trusted third party called the auctioneer to run the auction procedure correctly. In practice, finding such a trusted third party is challenging. Without such a trusted third party, several concerns arise. The main concern in running auctions is the correctness of the auction result presented by the auctioneer. While not a problem in open auctions, as participants witness the procedure, the problem arises in online auctions, which are gaining popularity [2]. Here, a malicious auctioneer has opportunities to manipulate the result without leaving a trace. For example, a malicious auctioneer can ignore the bids of specific participants, which manipulates the auction result. Furthermore, auctioneers learn the bids of all participants. They can exploit this knowledge and participate in insider trading or leak information to colluding entities [3].

Hence, both correctness and privacy play vital roles in auctions. Due to the lack of unconditional trust in the auctioneer, theoretical properties such as truthfulness may not hold in the real world. Overall, these concerns harm the market and competition. The field of research in auctions regarding privacy, correctness, and other desirable properties is called *privacy-preserving auctions*. We aim to investigate the issue of privacy and correctness and provide a solution for privacy-preserving double auctions.

1.1. Double Auctions

In *double auctions*, buyers and sellers submit bids and asks, representing how much of an item they are willing to buy or sell, respectively, at a specific price [4]. Double Auctions are used in trading commodities or shares of companies and are continuous or periodic. Continuous double auctions aim to find a match between a buyer and seller. When a buyer submits a buy offer, the auctioneer checks whether the buy offer is higher than the current lowest sell offer and vice versa for a newly submitted sell offer. If the auctioneer finds a match, the trade executes. Hence, a trade occurs between two parties if the buy offer is at least the sell offer.

In periodic double auctions, the auction procedure executes all trades at a specific time or after some time has passed. Such a double auction has an auction window during which participants are allowed to submit offers. At the end of the auction period, the auctioneer aggregates all bids and asks into demand and supply curves. By the law of supply and demand, as price increases, buyers are willing to buy less, and sellers are willing to sell more [5]. Hence, there will be a point where the supply curve meets the demand curve. The price at this point is called the market-clearing price. Bids at the market-clearing price or higher trade their items with asks at the market-clearing price or lower. Depending on the domain, the timespan of the auction window spans from a few milliseconds to multiple days.

In this work, we focus on periodic double auctions using a Walrasian mechanism [6]. In a Walrasian mechanism, buyers and sellers specify quantities at multiple prices, indicating their willingness to buy or sell. In theory, traders should specify quantities to buy or sell at all prices in a Walrasian mechanism. However, in reality, traders are only interested in prices where the quantity to buy or sell changes; hence, it is simpler to specify points where the quantities change. These kinds of double auctions are used in electricity exchanges across the globe [7]. We note that a popular double auction mechanism is for bids with only a single quantity of an item [8]. However, the double auction we are interested in allows trading multiple homogenous items simultaneously. These are double auctions where buyers and sellers simultaneously specify multiples of the same item, such as electricity or company shares.

Indeed, the Walrasian mechanism inherits similar problems as general auctions. Participants have to trust the auctioneer to run the auction correctly. At the same time, the auctioneer learns all offers submitted by participants, which leaks sensitive information to the auctioneer. These concerns are not as theoretical as one may hope.

1.2. Concerns with Auctions

Auctions suffer from a maliciously acting auctioneer when implemented in the real world. We present real-world examples where the auctioneer harmed the correctness and privacy of offers. These examples motivate the need for a privacy-preserving and verifiable double auction.

1.2.1. Correctness Concerns

Auctioneers have a unique position in the market. They learn the offers of participants and are trusted to compute the auction result correctly. However, auctioneers can manipulate the auction result. For example, a malicious auctioneer may inject offers of colluding participants after the auction closes or ignore participants' offers, thus manipulating the result. This kind of manipulation is hard to prove.

To motivate the correctness problem, we examine current allegations regarding the advertisement exchange program of Google [9]. Google runs a second-price sealed-bid auction, where the highest bid wins and pays the bid of the second-highest bidder. This setting is slightly different from double auctions but is adaptable to them. Google allegedly used its position in the market, telling buyers one price to pay while telling sellers a different lower price they receive. These prices were different than dictated by the procedure used. Google allegedly used this difference for other purposes in its advertisement ecosystem, increasing its profitability. Double Auctions are susceptible to the same problem. The auctioneer may demand buyers to pay a higher price than what the actual result would dictate and award sellers with a lower price they receive.

Procurement bidding highlights the correctness concern in auctions. Procurement bidding is a bidding process typically using sealed-bid auctions to award contracts from the government to, e.g., construction companies. Companies compete with each other on who receives the contract based on each company's conditions, such as price. In Procurement bidding, governments act as auctioneers. According to OECD, procurement bidding is one of the most corruption-prone activities of governments [10]. A cheating company may bribe the government to declare a different auction result than the actual result, thus manipulating the result and harming the market and competition. It is difficult to assess the damage such corruption causes. However, estimates are that "...10-30% of the investment in publicly funded construction projects may be lost through mismanagement and corruption..." [10].

This concern calls for solutions that allow verification of the auction result by the participants. Verification allows participants to trust the auction result declared by the auctioneer, combat manipulated results, and avert corruption.

1.2.2. Privacy Concerns

In typical auction systems, participants send their offers to a central auctioneer. The auctioneer then inspects all bids and asks in plaintext. While this allows computing the auction result, it enables the auctioneer to create profiles of all participants. Participants may consider bid information to be trade secrets, which would leak sensitive information to the auctioneer. The auctioneer may leak this information to other participants in the auction to give them a competitive advantage against competitors. The auctioneer may also use this information as insider knowledge and utilize it to their advantage or discriminate against participants.

To motivate the problem of privacy, we again examine current allegations. EPEX Spot provides an auction system to trade electricity using a double auction in Europe. Their intra-day auction provides a way to trade electricity in the short term, which is crucial for renewable energies. The European Commission has opened an investigation into EPEX Spot due to anti-competitive behavior [11]. According to the investigations, EPEX Spot limited the ability of (customers of) competitors to participate in the electricity exchange. EPEX Spot could limit their competitors since they could identify participants by the data present when participants submit offers. Due to the lack of privacy, EPEX Spot could thus manipulate the auction result.

The privacy concerns of auctions also get highlighted in work conducted by Bogetoft et al. [12]. In Denmark, sugar beet contracts get traded between farmers in a double auction, similar to the one we study. The question arose of who plays the role of the auctioneer. A natural choice was Danisco, the buyer of sugar beets in Denmark. However, this was not acceptable to farmers, as Danisco would gain private economic information. A survey by the authors questioned sugar beet farmers on how important bid confidentiality is to them. Indeed, 78% of farmers agreed that bid confidentiality is important to avoid revealing their economic position, which farmers feared Danisco could abuse.

This concern calls for solutions that preserve participants' privacy regarding their offers while not damaging economic properties. Pseudonymity or anonymity protects the identity of traders and would prevent such discrimination. However, the specific offers a trader submits or patterns in offers may also identify them. Thus confidentiality of offers is also a desirable property.

1.3. Use Case: Electricity Trading

The use case we present for periodic double auctions is electricity trading. Electricity producers such as solar panels or coal factories (sellers) trade with electricity consumers such as manufacturing plants or smart homes (buyers). Currently, electricity trading occurs on central platforms in Europe such as *EPEX Spot* [13] and *Nordpool* [14]. While different procedures exist to trade electricity on these platforms, the most significant volume of electricity trading occurs through *Day-Ahead* auctions. In Day-Ahead auctions, buyers and sellers submit offers for trading electricity the next day. Buyers and sellers submit bids and asks for each hour of the next day or submit blocks of bids and asks. We particularly focus on submitting offers for each hour.

The current system works as follows: Before the auction period ends, buyers and sellers submit several quantity-price pairs representing quantities they are willing to buy or sell at different prices in plaintext. The auction window closes every day at noon for the next day, and the range of possible prices is between -500€/MWh and 3000€/MWh. The auctioneer receives the quantity-price pairs of each trader and aggregates them into demand and supply curves. Then, the auctioneer computes the intersection point between supply and demand, which is the optimal price for trading [13]. Finally, the auctioneer broadcasts this price to all participants, declares the winners, and engages in the electricity transfer. There are a few differences in the procedures between Nordpool and EPEX Spot [7]. Nordpool uses linear interpolation between two points to compute the quantities for prices not mentioned in the offer. Meanwhile, EPEX Spot uses stepwise curves, assuming the same quantity until the next quantity-price pair, which is slightly more economically efficient [7].

The previously presented concerns for auctions also hold for electricity trading. The correctness of the auction result is an important property to ensure a fair electricity price. Electricity is not only a commodity to trade but also essential for ensuring efficient utilization of renewable energy sources [11] and hence vital for the future. REMIT [15] is a regulation implemented by the European Union to ensure fair competition in the electricity market. While strict regulations are in place, investigations suggest

that regulations may not have been satisfied [11]. We argue that a privacy-preserving and verifiably correct scheme ensures that electricity exchanges correctly follow regulations such as REMIT. Such a scheme will ensure that all market participants act with the same information to prevent insider trading and ensure that the auctioneer did not manipulate the auction result. Both privacy and verification of results help in this endeavor.

We note that the work in this thesis is adaptable to other domains. Dark pool trading [16, 17] is another domain with concerns such as privacy and correctness that researchers study. However, any domain using a general periodic double auction mechanism is a potential use case, such as emissions trading. In the real world, Partisia already offers a secure platform for sugar beet contract exchanges in Denmark [18]. Their auction mechanism ensures that a central auctioneer cannot learn participants' bids. This existing solution shows the potential for privacy-preserving and verifiable exchanges in the real world.

1.4. Research Questions

We focus on periodic double auctions using a Walrasian mechanism in the following work. We aim to design a privacy-preserving periodic double auction system that resembles those used in electricity exchanges. Due to the previously mentioned concerns, a solution should maintain the confidentiality of offers and allow for verification. We also add additional properties such as non-repudiation to ensure that a trader cannot deny having made an offer. Furthermore, electricity exchanges (and typical periodic double auctions) allow traders to withdraw offers before the auction period ends. We aim to allow such functional properties in the scheme. Lastly, economic efficiency should stay unaffected. Overall, the research question is as follows:

How can we construct a privacy-preserving verifiable double auction with confidential and undeniable offers that closely resembles the current auction procedure?

To answer the research question, we define the following sub-questions:

1. How can an auctioneer aggregate bids and asks and utilize them to compute the market-clearing price securely while preserving confidentiality?
2. How can the public verify the double auction result?
3. How can malicious behavior of participants be detected?

1.5. Contribution

Current works on privacy-preserving double auctions lack properties. Schemes that consider both confidentiality and verification require the participation of all entities. To the best of our knowledge, there exists no solution that closely resembles the current auction procedure of electricity exchanges in a privacy-preserving and verifiable manner. We aim to provide a scheme that incorporates confidentiality of offers and verification of results while allowing for functional properties such as withdrawal and preserving economic efficiency.

1.6. Outline

The thesis is structured as follows: In Chapter 2, we present an overview of knowledge and techniques required to understand the contributions of this thesis and related works. In Chapter 3 we present existing works on auctions and privacy-preserving double auctions schemes. In Chapter 4 we present our scheme, a privacy-preserving & verifiable double auction tailored to electricity trading. We evaluate our design in Chapter 5 from a theoretical and practical point of view. Finally, in Chapter 6 we conclude this work by discussing the scheme, re-evaluating our research questions, and providing future research directions.

2

Preliminaries

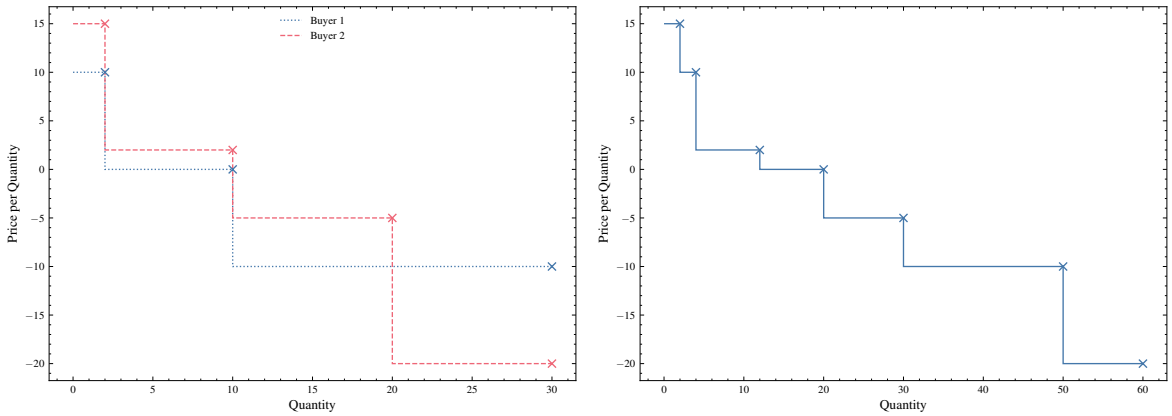
This chapter outlines the knowledge and techniques used in this work and related works of privacy-preserving auctions. We focus on knowledge about double auctions and cryptographic tools and present examples, diagrams, or short derivations of properties where appropriate.

2.1. Auctions Background

This section presents necessary knowledge about the theory behind double auctions and privacy-preserving auctions. We establish the general procedure in double auctions and properties in privacy-preserving auctions.

2.1.1. Double Auctions

Double Auctions come in many different flavors. We focus on double auctions where bidders and sellers bid on different quantities of the same item at distinctive prices, and the auctioneer evaluates the auction periodically. The auction uses a Walrasian mechanism [6] that allows bidders and sellers to bid their demand or supply at different prices. In theory, bidding on all possible prices is possible in a Walrasian mechanism. However, electricity exchanges allow traders to only submit offers on prices that change the quantity they want to buy or sell [7]. The quantity-price pairs that traders submit form a stepwise curve, which is the same approach EPEX Spot takes [7].



(a) Stepwise curves of two buyers submitting (quantity, price): Buyer 1: [(2,10), (10, 0), (30, -10)], Buyer 2: [(2, 15), (10, 2), (20, -5), (30, -20)]

(b) Aggregated demand curve of previous bidders described by: [(2, 15), (4, 10), (12, 2), (20, 0), (30, -5), (50, -10), (60, -20)]

Figure 2.1: Example offers of traders and aggregation of offers

Periodic double auction schemes using a Walrasian mechanism allow bidders and sellers to specify quantities they are willing to buy/sell at different price points. Each participant provides several quantity-

price pairs to the auctioneer, representing the amount a participant wants to buy/sell at these prices. Using these price-quantity pairs, the auctioneer can visualize the demand/supply of that participant graphically using a stepwise curve or staircase function, as depicted in Figure 2.1a, read from left to right. Due to the laws of supply and demand [5], economically rational buyers are more willing to buy low-priced products. In contrast, economically rational sellers are more willing to sell high-priced products. This concept holds as both groups want to maximize their profit in the auction. Due to this, as the price increases, buyers are willing to buy less, and sellers are willing to sell more. The graphical representations reflect this tendency.

Aggregating individual stepwise curves results in overall supply and demand curves. Figure 2.1b presents how two individual stepwise buy curves aggregate to an overall demand curve. Repeating the same procedure for sell offers yields an overall supply curve. To find the market-clearing price, the auctioneer finds the point of intersection of supply and demand. Graphically, the auctioneer overlays the supply and demand curves, as depicted in Figure 2.2. The price at the intersection point is called the market-clearing price, and the quantity is the quantity that buyers and sellers will trade. Buyers who want to buy at the market-clearing price or higher get to trade their products with sellers who sell at the market-clearing price or lower. Each buyer pays the market-clearing price, and each seller receives the market-clearing price. Figure 2.2 presents an example: All buyers buy (overall 20 items) at -3, but sellers only sell 20 items while offering 25 overall at this price. Not all traders get to trade their products, which is a feature of the double auction mechanism.

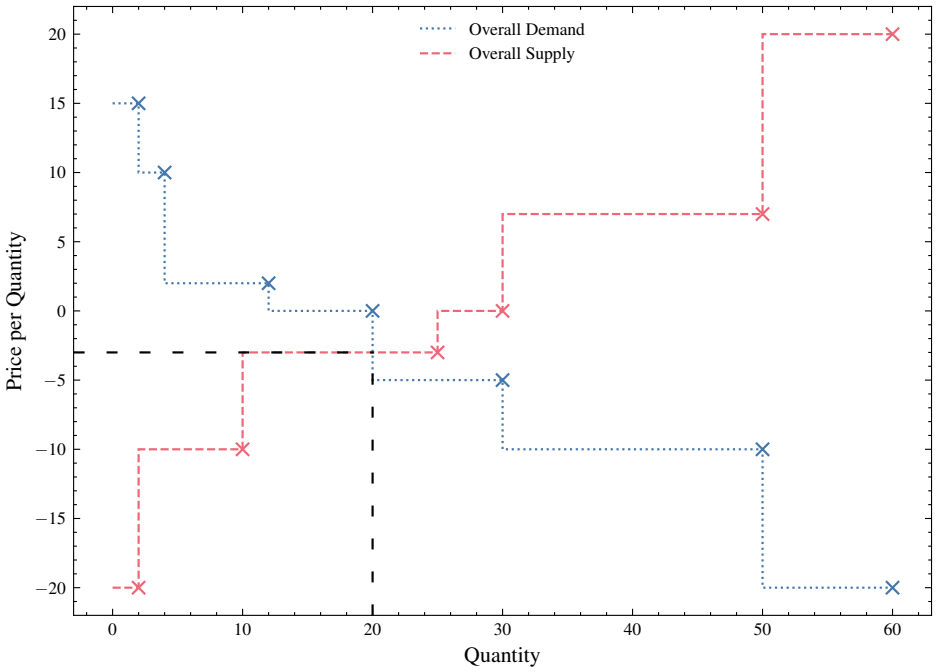
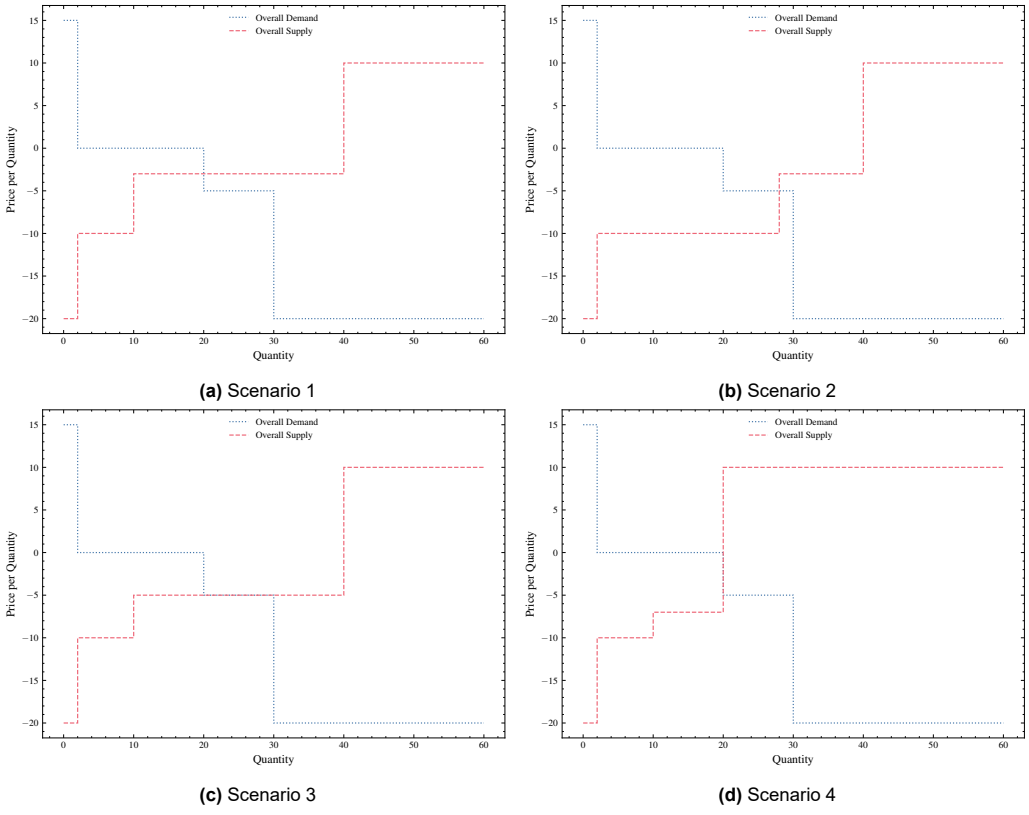


Figure 2.2: Intersection between supply and demand curve. The supply curve is described by: [(2, -20), (10, -10), (25, -3), (30, 0), (50, 7), (60, 20)], The demand curve is unchanged. The market-clearing price is -3, and the quantity traded is 20

Building the two curves and finding the market-clearing price by overlaying them is the easiest method, as finding the intersection is simply reading the graph. However, we note that finding the intersection is possible without drawing the graph by comparing demand and supply at different price points. A linear search finds this point by comparing demand and supply at ascending prices and noting when supply becomes higher than demand. However, due to the ascending/ascending quantities for demand and supply, a binary search is possible and more efficient in finding the intersection.

Figure 2.3 graphically presents four scenarios of the intersection between demand and supply. In scenarios 1 & 2, there is only one intersection point between demand and supply. However, in scenarios 3 & 4, there are multiple intersection points as the line segments intersect in the same orientation (horizontal or vertical at the same price or quantity, respectively). One solution is to choose the lowest price possible in scenario 3 and the highest quantity possible in scenario 4, which is the solution employed

by electricity exchanges [7] and thus also our solution. We note that more scenarios are possible for scenarios 3 & 4, depending on whether demand or supply changes first. However, this is not relevant for pricing. We also note that one of the assumptions is that buyers and sellers indeed act economically rational regarding their offers and follow the laws of supply and demand. Hence, the stepwise buy and sell curves are monotonically decreasing when buying and increasing when selling. Finally, we note that demand and supply may never intersect, which is rare [14].



(a) Scenario 1 **(b)** Scenario 2

(c) Scenario 3 **(d)** Scenario 4

Figure 2.3: Different scenarios of supply and demand intersecting

2.1.2. Properties in Privacy-preserving Auctions

Since the introduction of privacy-preserving auctions, various works have formalized the potential properties of privacy-preserving auctions [19–21]. These hold in general for all kinds of auctions:

- **Unforgeability**: A participant cannot impersonate another user. In this context, a trader cannot create offers for other traders other than themselves
- **Anonymity**: There is no link between the identity and the offer submitted by a trader
- **Pseudonimity**: A trader is identifiable by an identity (pseudonym) created, but no one can uncover the real identity of a trader given their pseudonym
- **Collusion-resistance**: The auction shall be resistant against collusion between the auctioneer and traders
- **Traceability**: Traders who have won the auction must be identifiable and traceable
- **Public verifiability**: Any entity is able to verify the final result of the auction
- **Non-repudiation**: After traders submit offers, they cannot deny having made the offer
- **Bidding-unlinkability**: It is unfeasible to create a profile of a trader given the submitted offers
- **Confidentiality**: The auction reveals the winners and their winning offers, but all other offers are kept hidden

2.2. Cryptography Background

This section presents background knowledge on various cryptographic techniques used in many of the works in privacy-preserving auctions. We also use some of these techniques. We assume some background knowledge of the mathematics behind cryptography, such as number theory or finite fields. We refer to background material for this knowledge [22].

2.2.1. Public-key Cryptography

Overview

Public-key encryption provides methods to encrypt and decrypt messages using pairs of keys. Each key pair consists of a public key pk_a and a secret key sk_a, bound to a party a. The keypair owner publishes their public key but should keep their secret key hidden. To encrypt a plaintext message m for entity a, we utilize an encryption algorithm $c \leftarrow Enc_{pk_a}(m)$ that outputs a ciphertext c. Entity a decrypts the ciphertext as $Dec_{sk_a}(c)$, using a decryption function and their secret key to get back the original message m.

An essential property of public-key encryption is the relationship between the public and secret keys. The secret key often generates the public key. However, the public key does not provide information about the secret key. This relationship is called a one-way (trapdoor) function, which is easy to compute in one direction but hard in the other direction. Indeed, public-key encryption methods use this idea and base their security on instances of one-way trapdoor functions. Examples include factoring or the discrete logarithm problem. For example, the discrete logarithm problem states that given a generator g of a group G of prime order q, finding x such that $g^x = y \pmod q$ for a public y is hard [22]. Note that the reverse direction is easy: Given x and g, it is easy to compute g^x and find y by repeated multiplications. The group G often uses cyclic groups of prime order q or elliptic curves.

Based on the discrete logarithm, the decisional Diffie-Hellman assumption (DDH) is defined as the following: Given g^a, g^b for some random $a, b \in \mathbb{Z}_q$ and suitable generator g in Group G with prime order q, one cannot distinguish between g^{ab} and g^c for a random $c \in \mathbb{Z}_q$.

Elliptic-curve Cryptography

Elliptic curves are a special type of mathematical curve that follow the equation:

$$y^2 = x^3 + ax + b \pmod q, \tag{2.1}$$

for integers a, b and prime number q. Carefully chosen Elliptic curves provide groups in which the DDH assumption holds and thus allow for versions of cryptographic protocols based on the DDH assumption. Compared to factoring-based or finite-field solutions, elliptic-curve cryptography has a significant advantage in key sizes required for a secure scheme due to the complexity of the best-known attacks. In particular, for 128-bit security, NIST recommends key sizes of 256-bits for elliptic curve cryptography, while factoring-based or finite field cryptography requires 3072-bit keys [23].

One fast elliptic curve used in this work is `Curve25519`, described by the equation:

$$y^2 = x^3 + 486662x^2 + x \pmod{2^{255} - 19}. \tag{2.2}$$

Indeed, in Curve25519, the DDH assumption holds for the generator defined by the basepoint.

2.2.2. Multiparty Computation

Multiparty computation (MPC) is a cryptographic technique that allows multiple parties to compute a function $f(x_1, ..., x_n)$ over their inputs x_i without revealing their inputs, nor with the help of a trusted third party. It aims to replace the role of a trusted third party with cryptographic techniques. Secret sharing, garbled circuits, and homomorphic encryption are all cryptographic techniques that enable MPC. This thesis uses homomorphic encryption, but we will shortly outline all techniques.

Secret Sharing

Secret sharing is a cryptographic technique that splits secret input into multiple shares that, when combined, recover the secret. Each user splits their secret into multiple shares and distributes them across multiple servers that compute on their received shares. Secret sharing techniques are often denoted by (t, n), where n refers to the number of shares created, and t refers to the minimum number of shares needed to recover the original secret. The number of shares created depends on the number of servers for computations.

Linear secret sharing enables MPC. Each server gets multiple shares and carries out linear computations on their shares. In the end, the servers combine their shares, and the final result is declared, which results in the secrets with the operations applied [24]. Hence, MPC is achieved by secret sharing the input x_i across multiple servers. To learn $f(x_1, ..., x_i)$, each server applies the function f on the received shares. Combining the shares yields the function applied to the inputs.

Garbled Circuits

Garbled circuits is an MPC technique that works when two parties want to compute a function together, where each party holds a secret input x_i [25]. The technique assumes honest-but-curious participants. The function f is encoded into a boolean circuit, which one party obfuscates and the other executes. Let us focus on a single gate: The idea is for the obfuscator to create a symmetric key for each input in the circuit and for each possible binary value the inputs can take. The obfuscator creates a lookup table with the result of the gate for the possible values of the inputs and uses the two symmetric keys created for these entries to encrypt the output of the gate. The obfuscator then randomly permutes the rows and columns. Finally, the obfuscator sends the lookup table to the evaluator with the obfuscator's input key for their input. The evaluator knows their input but does not know the symmetric key created for it by the obfuscator. At the same time, the obfuscator should not learn anything about the input of the evaluator. *Oblivious Transfer* provides the building block for this task.

The Oblivious transfer allows the evaluator to receive the correct key without the evaluator learning anything about the other key, and the obfuscator learns nothing about the input. Thus, the evaluator receives the key and find the correct entry to decrypt.

The above construction for a single binary gate generalizes to non-binary gates and general circuits. For general circuits, we note that instead of returning the result of the gate in the lookup table, the obfuscator provides the key for the following input.

Homomorphic Encryption

Homomorphic encryption is a cryptographic technique that allows computations over encrypted data. Homomorphic encryption allows performing operations on ciphertext which directly translates to operations on the data in plaintext. As an example, one has two ciphertexts c_1, c_2 encrypted with the same public key. One can multiply these two ciphertexts together to get $c_3 = c_1 \cdot c_2$. Decrypting this will give m_3. Using an appropriate homomorphic cryptosystem, this corresponds to decrypting c_1 and c_2 and adding the result in plaintext. That is:

$$Dec_{sk}(c_1 \cdot c_2) = Dec_{sk}(c_1) + Dec_{sk}(c_2). \tag{2.3}$$

Homomorphic encryption also enables MPC. Users encrypt their inputs with a public key and send their ciphertexts to a server not holding the corresponding private key. The server then applies operations such as addition or multiplication on the ciphertexts. The resulting ciphertext is an encryption of the input when function f is applied, which the server sends to the keyholder for decryption. The result is the function f applied to the plaintexts.

Depending on the homomorphic encryption scheme, users can apply different functions f. *Partially homomorphic encryption* describes schemes where users can apply only one kind of operation on data, such as addition. Meanwhile, *fully homomorphic encryption* describes schemes where users can apply an arbitrary number of different operations on data, usually addition and multiplication. Partially homomorphic schemes are generally faster due to the computationally more complex procedure necessary for fully homomorphic schemes. We use partially homomorphic encryption schemes in this work.

Overview
All MPC techniques have particular use-cases and strengths and weaknesses. Secret sharing is particularly powerful since it provides security against computationally unbounded adversaries as long as a specific number of servers (usually the majority) is honest. However, secret sharing requires multiple non-colluding servers and interactivity between the servers.

Garbled circuits provide an elegant and secure method in the two-party case, as it relies on symmetric cryptography, which is fast for the needed operations. However, it only works in the two-party case and assumes honest-but-curious entities, not malicious ones. While it is possible to adapt the procedure for malicious entities, it becomes more computationally expensive.

Lastly, homomorphic encryption has the advantage that a single server is responsible for performing operations on the ciphertext, which saves interactivity. However, compared to secret sharing, homomorphic operations such as adding or multiplication are slower due to the required cryptographic operations. Furthermore, homomorphic encryptions have limits on the applicable operations.

We focus on homomorphic encryption, as it provides ways to compute on encrypted data in the client-server model, which is close to the existing auction model.

2.3. Cryptographic Techniques

The following section provides background on the cryptographic techniques used in our scheme.

2.3.1. Homomorphic Encryption

Paillier Cryptosystem
This work uses the Paillier cryptosystem as a partially homomorphic encryption scheme, providing homomorphic addition [26, 27]. The security of the Paillier cryptosystem relies on the hardness of the decisional composite residuosity problem, which is assumed to be hard [27]. The decisional composite residuosity problem states that there exists no polynomial-time algorithm to decide whether there exists a y, given a composite n and an integer z such that:

$$z = y^n \pmod{n^2}. \tag{2.4}$$

We illustrate the algorithms for key generation, encryption and decryption in the Paillier cryptosystem.

Key generation with k-bit security

1. Generate random prime numbers p, q independently of equal length $k/2$.
2. Compute $n \leftarrow pq, \lambda \leftarrow (p-1)(q-1), \mu \leftarrow \lambda^{-1} \pmod{n}, g \leftarrow n+1$
3. Public key $pk \leftarrow (n, g)$ private key $sk \leftarrow (\lambda, \mu)$. Publish the public key

Encryption of plaintext m

1. Ensure $0 \leq m < n$
2. Select random factor r uniformly from $0 < r < n$
3. Compute and publish $c \leftarrow g^m \cdot r^n \pmod{n^2}$

Decryption of ciphertext c

1. Define $L(x) \leftarrow \frac{x-1}{n}$
2. Compute $m \leftarrow (L(c^\lambda \pmod{n^2}) \cdot \mu) \pmod{n}$

The Paillier cryptosystem is additively homomorphic. Hence, the following relation holds:

$$\begin{aligned} Dec_{sk_a}(Enc_{pk_a}(m_1) \cdot Enc_{pk_a}(m_2)) &= Dec_{sk_a}(g^{m_1} r_1^n \cdot g^{m_2} r_2^n \pmod{n^2}) \\ &= Dec_{sk_a}(g^{m_1+m_2} \cdot (r_1 r_2)^n \pmod{n^2}) \\ &= m_1 + m_2. \end{aligned} \tag{2.5}$$

We also note that the Paillier cryptosystem is semantically secure. Hence, the encryptions of the same message are not equivalent due to the randomness used.

Exponential Elliptic-curve-ElGamal (ECEG)
This work also uses Elliptic-curve-ElGamal in the exponent. The idea is to encode messages in the exponent rather than in the base, as done in standard ElGamal [28]. The difference between standard Elliptic-curve-ElGamal and Exponential Elliptic-curve-ElGamal is only in the encryption and decryption function. This construction creates an additive homomorphic scheme instead of a multiplicatively homomorphic scheme. Using Elliptic-curves makes operations such as exponentiations faster due to smaller key sizes at the same level of security. However, one problem of ECEG is that a message must be encoded using a generator. Decoding the message requires either brute-forcing, rough knowledge of the range of the message, or breaking the discrete logarithm problem, which restricts the use cases. ECEG is particularly fast to check if a ciphertext is an encryption of zero, called a *Zero-check*. We outline the key generation, encryption, decryption, and zero-check.

Key generation

1. Pick suitable elliptic curve such as `Curve25519` with generator G
2. Select random factor r from $0 < r < 2^{255}$
3. Private key $sk \leftarrow r$, public key $pk \leftarrow sk \cdot G$

Encryption of plaintext m

1. Select random scalar y from $0 < y < 2^{255}$
2. $c_1 \leftarrow yG$
3. $c_2 \leftarrow mG + y \cdot pk$
4. Publish (c_1, c_2)

Decryption of ciphertexts (c_1, c_2) to mG

1. Compute $c_2 - c_1 \cdot sk = mG + y \cdot pk - yG \cdot sk = mG + y \cdot pk - y \cdot pk = mG$

Zero-check

1. Compute $mG = c_2 - sk \cdot c_1$
2. Return `True` if $mG = 0 \cdot G$, else `False`

Exponential Elliptic-Curve ElGamal is additively homomorphic. Hence, the following relation holds:

$$\begin{aligned} Dec_{sk_a}(Enc_{pk_a}(m_1) \cdot Enc_{pk_a}(m_2)) &= Dec_{sk_a}((y_1G, m_1G + y_1 \cdot pk) \cdot (y_2G, m_2G + y_2 \cdot pk)) \\ &= Dec_{sk_a}(y_1G + y_2G, m_1G + m_2G + (y_1 + y_2)pk) \\ &= (m_1 + m_2)G. \end{aligned} \tag{2.6}$$

2.3.2. Cryptographic Commitments

General
A cryptographic commitment, denoted by C_m, is a message allowing senders to commit to a specific message m but hide the contents of m for now. Later, the sender reveals m (and some extra information), and anyone who received this information and the commitment can verify whether the revealed message corresponds to the original commitment. Hence, commitment schemes consist of a commit step, a revealing step, and a verify step. The main use of commitment schemes is to set something and reveal it later. As an example, commitment schemes are useful in auctions. A trader first commits to a specific offer. Only after the auction period ended the trader reveals the offer. The advantage here is that an auctioneer does not know the offer until after the trading period and thus cannot participate in insider trading. Commitment schemes have two properties:

- **Hiding**: Given a commitment C_m, it is hard to compute the message m in the commitment
- **Binding**: Given a commitment C_m on a message m, it is hard to find a different message m' (and parameters) such that $C_m = C_{m'}$

The Hiding and Binding properties have two notions of security:

- **Computational:** A computationally unbounded adversary can break this property. These properties rely on mathematically hard problems such as the discrete logarithm problem
- **Information-theoretic:** Even a computationally unbounded adversary cannot break this property

Pedersen Commitments
Pedersen commitments use the discrete logarithm problem as the underlying hard problem [29]. We illustrate the parameter generation, commitment, and verification process:

Public parameter generation for λ-bit security

1. Pick group G with generator g with prime order q of bit-length λ
2. Select generator h as a random element from G
3. Release (g, h) to use as public parameters for commitments

Commit to message m

1. Ensure $0 \leq m < q$
2. Select random factor s uniformly from $0 \leq s < q$
3. Compute and publish $C_m = g^m \cdot h^s \pmod q$

Verify Commitment given revealed (m', s')

1. Verify $C_m = g^{m'} h^{s'} \pmod q$

The Pedersen commitment is a computationally binding and information-theoretic hiding commitment scheme. In this case, computationally binding means that an adversary who created a commitment on message m with randomness s has to solve the discrete logarithm problem to find different parameters $(m', s') \neq (m, s)$ that generate the same commitment. Cryptographers believe that solving the discrete logarithm problem is hard and thus requires considerable computational power to accomplish in a reasonable time. Information-theoretic hiding means that no adversary can find the message m of the commitment. An intuitive reason is because of the randomness in the commitment. Hence, there are other pairs $(m', s') \neq (m, s)$ that generate the same commitment, and thus an entity who found a suitable pair (m', s') cannot be sure that the values found are correct.

Furthermore, Pedersen commitments are additively homomorphic:

$$C(m_1, s_1) \cdot C(m_2, s_2) = h^{m_1} \cdot g^{s_1} \cdot h^{m_2} \cdot g^{s_2} = h^{m_1+m_2} \cdot g^{s_1+s_2} = C(m_1 + m_2, s_1 + s_2). \tag{2.7}$$

Hence the multiplication of two commitments on messages m_1, m_2 creates a valid commitment on the message $m_1 + m_2$. To verify this commitment, the original senders would need to release $s_1 + s_2$.

Cryptographic Hash Functions
General
In general, hash functions are functions which map an input of any kind of length to a constant-sized output, often called a hash or digest. Hash functions are useful in many different scenarios to provide an output of the same format. For cryptographic purposes, cryptographic hash functions are particularly important as they provide a one-way function, typically denoted by $H(\cdot)$. It is easy to compute the hash of an input, but it is hard to compute the input that created a certain hash. Crypographic hash functions have three properties:

- **Pre-image resistance:** Given a hash h, it is hard to recover a message m such that $H(m) = h$
- **Second-pre-image resistance:** Given a message m_1, it is hard to find a message m_2 where $m_1 \neq m_2$, such that $H(m_1) = H(m_2)$
- **Collision resistance:** It is hard to find messages m_1, m_2 where $m_1 \neq m_2$, such that $H(m_1) = H(m_2)$

Given these properties and the idea behind the pigeonhole principle, we note that finding the same hash for two different messages is possible. However, breaking any of the above properties for a cryptographically safe hash function should be hard. An example of a cryptographic hash function is SHA-256.

2.3.3. Cryptographic Signatures

General

Cryptographic signatures are a kind of digital signature on a message using public-key cryptography. The idea is for the signer to carry out a signing procedure on a message using their private key. Given the signature and the message, anyone with the signer's public key can verify that the signer signed the message. Cryptographic signatures provide properties such as non-repudiation or integrity of the message when sending them. For example, a signer signs a message and sends it to the receiver together with the message. Given the signer's public key, the receiver can verify that the message was not tampered with by verifying the signature on the received message.

Ed25519

Ed25519 is a digital signature algorithm on `Curve25519` based on twisted edwards curves [30]. In essence, they use carefully chosen parameters and elliptic curves for a variant of Schnorrs signature [31]. We outline the general procedure for Schnorrs signature:

Parameter generation

1. Choose a suitable group G of prime order q with generator g
2. Generate a random signing key x
3. Compute public key $y \leftarrow g^x$ and release it

Sign a message m

1. Choose random k from $0 < k < q$
2. Compute $r \leftarrow g^k$
3. Compute $e \leftarrow H(m||r)$
4. Compute $s \leftarrow k - x \cdot e$
5. Release (s, e) as signature

Verify signature (s, e) **is a signature on message** m

1. Compute $r' \leftarrow g^s y^e = g^{k-xe} g^{xe} = g^k$
2. Compute $e' \leftarrow H(m||r')$
3. Verify whether $e = e'$

Ed25519 offers fast key generation and signing compared to factoring-based digital signatures algorithms like RSA. However, the verification of signatures is slightly slower.

Blind Signatures

Blind signatures are a cryptographic technique to get a signature on a message while the signer stays oblivious to the contents of this message. The difference between standard cryptographic signatures and blind signatures is that the signer can view the message to sign in standard signatures. Meanwhile, in blind signatures, the signer cannot view the message to be signed as it is blinded. Electronic voting presents a use case for blind signatures. It may be essential to sign a voter's ballot to verify its authenticity in voting. However, the signing authority should not learn the vote of a voter.

RSA (Blind) Signatures
RSA is a cryptographic scheme that presents practical algorithms for encryptions and digital signatures. We illustrate the key generation, signature, and verification algorithm.

Parameter generation for λ-bit security

1. Generate distinct prime numbers p, q of length $\lambda/2$ to be kept secret
2. Compute $n \leftarrow p \cdot q, \phi(n) \leftarrow (p-1)(q-1)$
3. Generate e such that $gcd(\phi(n), e) = 1$
4. Compute $d \leftarrow e^{-1} \pmod{\phi(n)}$
5. Public key $pk \leftarrow (e, n)$, Private key $sk \leftarrow (d, n)$. Publish public key

Sign an integer message m

1. Compute signature $sig_m \leftarrow (m)^d \pmod{n}$

Verify signature sig_m is a signature on integer message m

1. Verify $(sig_m)^e = ((m)^d)^e \pmod{n} = m$

A signer can use a hash function to generate an integer representation of the message. A signer thus use RSA to sign any integer message using their secret key. However, the signer can also view the message's contents. RSA can be adapted to create a blind signature protocol on a message sent by a sender A. Protocol 2.1 presents how to create a blind signature.

Sender A		**Signer** B
$(e, n) \leftarrow$ public key of signer		$(d, n) \leftarrow$ private key of signer
$m \leftarrow$ message to be signed		
$r \leftarrow_{\$} \mathbb{Z}_N$		
$m' \leftarrow mr^e \pmod{n}$		
	$\xrightarrow{m'}$	
		$s' \leftarrow (m')^d \pmod{n}$
		$= (mr^e)^d \pmod{n}$
		$= m^d r \pmod{n}$
	$\xleftarrow{s'}$	
Recover $s \leftarrow s' \cdot r^{-1} \pmod{n} = m^d \pmod{n}$		

Protocol 2.1: Blind signature protocol

To get a blind signature on the message m, the sender A first blinds the message m by computing the value $m' \leftarrow mr^e \pmod{N}$ where r is a random blinding factor and e is the public key of the signer B. A then sends m' to B, who signs this blinded message to get: $s' \leftarrow m'^d = m^d r \pmod{n}$, and sends it back to A. A now recovers from s' a signature s on m, by performing $s \leftarrow s' \cdot r^{-1} \pmod{n}$, to cancel the randomness. Given s' or m' the signer has no information about the contents of the message, since they do not know the randomness r used.

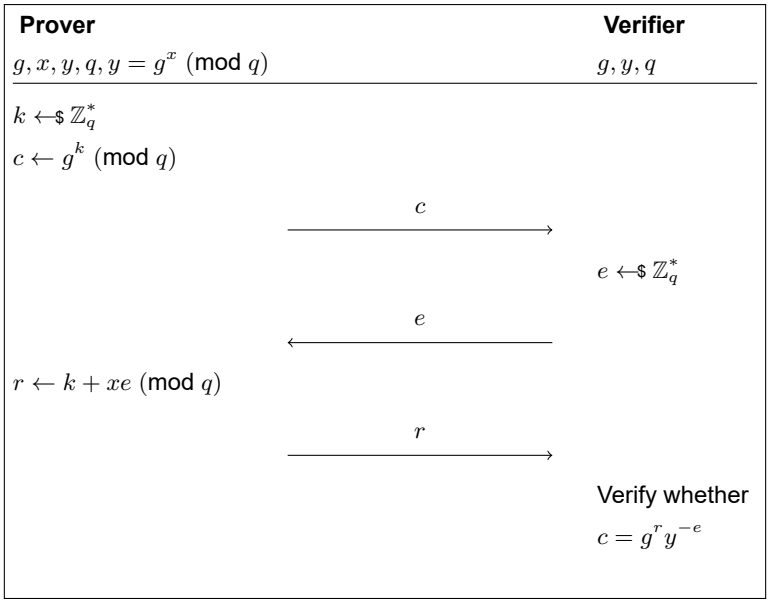
2.3.4. Zero-knowledge Proofs

General
Zero-knowledge proofs are a kind of cryptographic proof that a prover creates for a verifier to verify a particular statement, where the verifier gains no information about the underlying inputs. For example, it

is possible to prove to a verifier that a number is in a specific range without revealing the actual number but only a commitment of the number. While sounding impossible, there are real-life analogies that create a zero-knowledge-proof such as Ali baba's cave or the colorblind game [32]. Multiple protocols are feasible for zero-knowledge-proofs. The most versatile proofs are the likes of zk-SNARKS/zk-STARKS or Bulletproofs, which prove arbitrary statements by encoding the statement into an arithmetic circuit. However, to prove cryptographic statements, Sigma protocols are a suitable alternative. Zero-knowledge proofs have the following properties:

- **Completeness**: If the statement to be proven is true, an honest verifier accepts the statement with a large probability
- **Soundness:** A cheating prover who wants to convince an honest verifier that a false statement is true can do so with a small (negligible) probability
- **Zero-knowledge:** Proving the statement does not reveal additional information to the verifier

Sigma Protocols
Sigma protocols are a type of zero-knowledge proof that allow proving statements with structures such as knowledge of a discrete logarithm. All Sigma protocols follow the same structure: a commitment step, a challenge step, and a response step. Protocol 2.2 presents an example of a Sigma protocol on the knowledge of discrete logarithm. The prover wants to prove knowledge of a secret value x to a verifier such that $g^x = y \pmod q$ for a public y without revealing x. In particular, we note that the first message c is the commit step, the message e is the challenge step, and the message r is the response step.



Protocol 2.2: Example of a sigma protocol for proving knowledge of x such that $g^x \pmod q = y$

Sigma protocols are an example of an honest-verifier zero-knowledge proof. Honest-verifier means that one requirement for the proof to be zero-knowledge is that the verifier does not behave maliciously in the protocol by, e.g., choosing challenges not random but carefully crafted. Furthermore, Sigma protocols require interactivity between the prover and the verifier.

Fiat-Shamir Heuristic
The Fiat-Shamir heuristic is a technique to transform an (honest-verifier) interactive zero-knowledge proof into a non-interactive zero-knowledge proof in the random oracle model [33]. The random oracle model assumes the existence of a truly random function to prove security [34]. The Fiat-Shamir heuristic uses cryptographic hash functions to act as random functions [1]. One of the requirements to use the

[1] It is slightly controversial whether hash functions are true random oracles, as attacks are possible. Nevertheless, it is a viable approach to prove security

Fiat-Shamir heuristic is that the zero-knowledge protocol is *Public coin*, which means that the verifier releases a random challenge to the prover.

The Fiat-Shamir heuristic replaces the challenge step sent from the verifier to the prover by computing a hash. Depending on the notion of security, different things need to be included in the hash [35]. In particular, it is essential to include the statement to prove in the hash computation to achieve a strong Fiat-Shamir transformation. Using the Fiat-Shamir heuristic allows any entity to be the verifier of the zero-knowledge proof [36]. Furthermore, the Fiat-Shamir heuristic allows any party to create a Zero-knowledge proof using the Sigma protocol without interaction with the verifier.

Bulletproofs
Bulletproofs [37] are a special construction to construct zero-knowledge proofs on any statement encodable into an arithmetic circuit. The idea behind Bulletproofs is an efficiency improvement to the inner product proof of Bootle et al. [38]. The main idea is to prove relationships between two vectors of commitments. The relationship between the two vectors is encodable into an arithmetic circuit, and the proof system proves the relationship in zero-knowledge. Bulletproofs rely on the hardness of the discrete logarithm problem.

Bulletproofs are especially powerful for range proofs that allow a prover to prove that their input is in a specific range without revealing the input. The input is in the form of a Pedersen commitment, and the proof is efficient in size. For more details on the construction of bulletproofs, we refer to the original work [37].

2.3.5. Cut And Choose
Cut and Choose is a technique used in the two-party case involving malicious participants. In a typical scenario, A wants to prove to B that messages are in a particular format. A generates λ messages of the same format and blind them. B chooses $\lambda-1$ messages that A will unblind. A sends the information to unblind the $\lambda - 1$ terms to B, which B uses to unblind the messages. If all $\lambda - 1$ messages are of the correct format, B trusts that the remaining message is also of the correct format. This technique works since B gets to choose which messages to uncover themselves. Furthermore, A cannot change the messages after sending them to B. In this context, λ is a security parameter that ensures that a malicious A does not get lucky and creates a message that is not legal but does not get selected by B. Hence, the choice of λ directly affects the scheme's security.

2.3.6. Public Bulletin Board
Bulletin boards in cryptography work similarly to physical bulletin boards. They allow anyone to append information onto a publicly accessible medium for anyone else to view [39]. A bulletin board allows for use-cases such as broadcasting information and ensuring that information was indeed published. We require the bulletin board to be append-only, so information is not deletable, but users can only add new information. We also require the bulletin board to record the time when publishing information by appending the timestamp when receiving a message. A trusted third party or multiple parties can maintain a public bulletin board, but it is important to carefully choose who controls the bulletin board. Blockchains are one way to realize a decentralized public bulletin board that only appends information. Real-life examples of bulletin boards are Google's Certificate Transparency project [40].

3

Related Work

Researchers in cryptography have thoroughly studied auctions, proposing various solutions. Many problems that auctions face are similar to problems studied by cryptographers, such as the millionaire's problem [41]. We classify schemes for privacy-preserving auctions into three categories that differ in who fulfills the role of the auctioneer.

The first category consists of schemes in which the role of the auctioneer spreads over multiple semi-trusted servers using secret sharing. Each server gets a share of the offer and invokes computations on their share to get a result. In the end, the servers combine their shares to obtain the final result. One property is that the security of these schemes does not rely on computational security but information-theoretic security if the servers act honestly. However, secret sharing requires multiple independent entities who do not collude or have an honest majority, which may be challenging to provide in practice. Furthermore, each entity carries out the same amount of work.

The second category consists of schemes in which the role of the auctioneer spreads over all participants, which is an extension of the first category. Participants have to communicate with each other to resolve the auction and identify the winner. One advantage is that participants have an economic incentive, to be honest, as they have bid in the auction. However, this technique requires interactivity between all participants in perhaps multiple rounds. Depending on the domain, interactivity between all participants may not be possible. Participants may go offline, lose connection, or do not have the resources to participate in the protocol.

In the third category are schemes where a single auctioneer is present, often helped by one or more agents who do not collude. The auctioneer receives all offers but must communicate with agents for specific computations. The advantage here is that participants are not involved in the computations, and each server has a specific role. However, collusion between the auctioneer and the agent makes these schemes insecure or leaks information. Our scheme will fit into this category to realize a privacy-preserving double auction protocol.

In this chapter, we explore works of all three categories, analyzing their contributions and the strengths and weaknesses of their approaches. We introduce and describe general influential auction schemes and existing works on privacy-preserving and verifiable double auctions. Since double auctions are a specific type of auction, we gain inspiration from the early works on auctions.

3.1. General Auctions

To study general auction schemes, we first consider existing surveys on auctions. Alvarez et al. [42] study different sealed-bid auctions, providing an overview of the existing schemes and some of the field's problems. Their survey reviews schemes for problems such as first-price and second-price sealed-bid auctions. It concludes that computational efficiency, the malicious model, and unconditional security are areas of improvement for sealed-bid auctions.

Franklin & Reiter [19] propose one of the earliest schemes on sealed-bid auctions using verifiable signature sharing. The scheme works by using several semi-trusted parties to solve the problem of first-price sealed-bid auctions, where the winner pays the amount they bid. Each bidder has a signature of their bank on the electronic money they possess. In the first phase, bidders submit a bid by providing signature shares of their electronic money to each server. Each server then verifies that the share they received is correct. Once the bidding period closes, the second phase begins. In the second phase, each server sends its share to the other servers to reconstruct the signature on the electronic money. Once they reconstruct the signature, the servers collectively determine the highest bidder, announce the winner and collect the money. A third of the servers may be malicious, and the scheme stays secure due to using a threshold signature sharing scheme. Their scheme makes sure that bids are only accessible once the bidding period closes and that the bid is not disputable. However, servers collectively learn the bids of all participants, thus not having any confidentiality. Furthermore, it is not possible to verify the auction result.

Suzuki et al. [43] propose a first-price sealed-bid auction scheme relying on hash functions. Their protocol uses a chain of hashes that resembles the idea behind blockchain. Let P denote the fixed set of possible prices. There are two possible messages to send: bid or not bid. Bidders who want to bid at price P_i prepare a chain of messages. The chain starts with a secret seed. The next element in the chain consists of the hash of the message to bid on the price P_i and the previous block's hash. Subsequent elements are constructed by hashing the message to not bid on the price and the previous block's hash. The bidder appends new blocks until reaching the highest price and releases the last element in the chain to the public. After the end of the auction period, the auctioneer starts from the highest price P_n and asks bidders to reveal the contents of block $n-1$. The auctioneer then checks whether any bidder bids on this price by checking for the two possible messages (bid or not bid) and verifies that the block n is correct. If no bidder bid, the auctioneer repeats the process for price P_{n-1}. Eventually, the auctioneer will find a price for which one buyer has bid. To remove the interactivity required, they propose a second method. The seed is secret-shared among multiple auctioneers, and the servers cooperate to find the first price at which a bidder bid. Their construction mostly relies on hash functions and is thus computationally efficient and allows for verification of the result. However, the first method relies on the interactivity of all participants.

Brandt [44] proposes a solution to the problem of Vickrey (second-price sealed-bid auctions) auctions using search algorithms. In Vickrey auctions, the highest bid wins and pays the price of the second-highest bid. The auctioneer first discretizes the possible prices before the auction. Bidders indicate their bid through an encrypted binary value in a column vector, where each row signalizes whether the bidder is willing to pay the price. Each bidder encrypts each row with a separate symmetric key which only the bidder knows, and sends their vector to the auctioneer. The auctioneer assembles a bid matrix consisting of the vectors sent by bidders as columns and publishes the matrix. Brandt describes various procedures for determining the market price by unmasking matrix values, each with advantages and disadvantages. The auctioneer follows the procedure requesting bidders to uncover specific value by releasing the key used. The protocol thus allows for verifying the auction result. However, one problem lies in the interactivity required. Participants must reveal their encryption keys to the auctioneer to unmask values and thus have to remain online for the duration of the protocol.

Król et al. [45] propose PASTRAMI, a privacy-preserving auction for heterogeneous items. PASTRAMI deals with assignment auctions, where the auction matches buyers and sellers of items. Buyers bid on specific items or any item that matches criteria and get assigned to at most one item. The protocol uses the blockchain and threshold blind signatures using multiple servers. Buyers and sellers go through two phases: In the commit phase, traders request blind signature shares from the servers, which traders unblind locally and combine to get a signature on their price. In the reveal phase, buyers and sellers reveal their signatures onto the blockchain by providing the blinding terms. Given all the signatures, a dedicated node computes the auction result according to a standard algorithm, which returns the assignments of buyers to items. Any entity can then verify whether this is an optimal assignment or not. In case of dispute, the entities send proof of the wrong assignment to a smart contract, which efficiently verifies the proof's correctness. PASTRAMI hides the price until the revealing phase; thus, there is no complete confidentiality of non-winning offers. Furthermore, the system only deals with single items, not multiples of the same item.

While not directly relevant to double auctions, these schemes provide insights into the design of privacy-preserving auctions. Many works distribute trust over multiple entities or let participants resolve the protocol. Newer schemes use smart contracts to automate malicious behavior detection, for example.

3.2. Double Auctions

Researchers from both economics and cryptography studied double auctions. While economists are concerned with the theoretical properties of double auctions, cryptographers are interested in providing solutions to the privacy and verification problems that double auctions face. We study the cryptographic works in more detail. We distinguish schemes for double auctions based on who fills the role of the auctioneer.

Work	Conf.	Public Ver.	Mult. Offers	Auctioneer	Malicious Auctioneer	Malicious Traders
Bogetoft et al. [12]	●	○	●	Servers	◐	●[1]
Cartlidge et al. [16, 17]	●	○	○	Servers	◐	○
Abidin et al. [46]	●	○	○	Servers	◐	○
Wallrabenstein et al. [47]	●	●	●	All	◐	●
Liu et al. [48]	●	●	●	All	◐	◐
Wang et al. [49]	●	●	○	All	●	●
Liu et al. [50] [2]	●	●	○	Single	●	●
Xu et al. [51]	●	○	○	Single	○	○
Sarenche et al. [21]	○	●	●	Single	●	●
Galal et al. [52]	○	●	○	Single	●	●
This work	●	●	●	Single	●[1]	●[1]

Table 3.1: Existing Double Auction schemes, 1 = Assuming economic rationality, 2 = Using Intel SGX on which attacks are possible

Table 3.1 presents protocols from existing works and their properties. Conf. (Confidentiality) refers to the confidentiality of offers against the auctioneer; Public Ver. (Public Verification) refers to whether individual participants are able to verify the final result of the auction. Mult. Offers (Multiple Offers) refers to whether the double auction allows submitting multiple offers. Auctioneer refers to who plays the role of the auctioneer in the protocol. Here, `Servers` refers to traders communicating with several servers that resolve the protocol. `All` refers to all traders cooperating to fulfill the role of the auctioneer. `Single` refers to traders communicating with a single server as the auctioneer and the auctioneer getting help for computations from other servers. Hence, one auctioneer receives all the offers, and an agent helps the auctioneer with computations. Malicious refers to whether the protocol considers and protects against malicious behavior of the auctioneer and traders. In the case of the auctioneer, we differentiate whether there are multiple auctioneers (Servers and All) or a single auctioneer (Single). If there are multiple auctioneers, we state how many servers need to be honest (roughly half or not). If there is only one auctioneer, we state whether they can be malicious or not. For inputs of clients, we differentiate based on whether the client input can be malicious or not. Economic rationale refers to the entities acting in their economic interest.

3.2.1. Auctioneer: Servers

Bogetoft et al. [12] propose multiparty computation for the double auctions we study. Their work demonstrates the practicality of secure multiparty computation when applied to secure double auctions for sugar beet trading. Their scheme uses verifiable secret sharing involving representatives of buyers, sellers, and the research project as the three servers. Participants submit bids and asks representing how much they are willing to buy or sell at all possible prices. The bids and asks are then secret shared among the three servers for aggregation. Each server verifies that their received share is correct by the verification property of verifiable secret sharing. The servers then aggregate the individual shares to construct demand and supply curve shares. The parties compute the market-clearing price using secure comparisons on secret shared values using a binary search. The secure comparisons work by computing the difference between the two values to compare, with some hiding factors, and computing the most significant bit. After a participant submits their offer, no interactivity is required, and participants

can submit multiple offers. However, the protocol does not allow participants to verify the results, and corrupting two out of three parties renders the protocol insecure.

Cartlidge et al. [16] propose a solution for dark pool trading based on secret sharing. Dark pool trading is a way to exchange stocks of companies while hiding orders from the public. Influential buyers and sellers use dark pool trading to ensure that a large volume of sell offers does not trigger large-scale sell offers from the public. Similar concerns as presented in Chapter 1 apply for dark pool trading [17]. Cartlidge et al. propose solutions for continuous, periodic, and volume-based double auctions. For their periodic double auction, their scheme proposes to insert new buy and sell orders at the end of two sorted lists and perform an insertion sort to sort the lists correctly with the cooperation of the servers. The servers then iterate over the lists and mark successful trades if the price of a buy order is higher than the price of a sell order. Due to secret sharing, their scheme preserves the confidentiality of offers. However, their procedure does not allow participants to verify results nor submit multiple offers simultaneously.

Cartlidge et al. [17] extend their scheme to make it more realistic for real-life dark pools. Keeping in mind the properties of a real-life dark pool called “Turquoise Plato”, they design a protocol to accommodate multiple heterogeneous items (company stocks) by providing a way to assign sets of servers to items while staying oblivious to the assignments. Furthermore, they extend their protocol to incorporate functional properties such as withdrawing offers. They show that privacy-preserving dark pool trading is a real use case given a few optimizations.

Abidin et al. [46] present a solution to trade electricity in a privacy-preserving way using multiparty computation. Their construction uses three servers controlled by independent entities but is extendable to more servers. Using secret sharing, traders distribute their offers over the servers, and the servers compute on the secret shares. Their scheme finds the market-clearing price by firstly sorting all offers, regardless of whether they are buying or selling. Then, the servers aggregate the total demand and match supply offers until matching demand. However, they assume a semi-honest majority of auctioneer servers. Furthermore, their work shows that sorting offers based on an encrypted price is expensive and takes most of the computation time, as it requires the participation of all servers. Hence, an approach that does not rely on sorting may perform better.

3.2.2. Auctioneer: All Traders

Wallrabenstein et al. [47] provide a privacy-preserving Walrasian auction, closely resembling the auction we study. The authors argue that the valuations of a trader (utility functions) leak sensitive information and should be kept hidden. Their protocol makes use of the Paillier cryptosystem. The basic protocol works by a seller initiating an initial price and all buyers adding on their demand at that price. The first buyer initializes the demand and sends it to the subsequent buyers, who add on their demand homomorphically. The final buyer checks the current round and determines whether another round is needed. If another round is needed, the final buyer computes the excess demand and the price update and sends the new price to the first buyer to restart the procedure for a new price. If the protocol reaches the last round, the final buyer sends the demand to the seller. The seller finalizes the protocol by decrypting the price and demand. The authors extend their scheme to consider the malicious behavior of traders by buyers committing to their utility function. However, their scheme only considers a single seller for an item and not multiple sellers selling the same item. Furthermore, communication between buyers is required as the coalition of buyers effectively emulates the role of the auctioneer. The interactivity may be expensive regarding communication costs as it may take some rounds to find the equilibrium price.

Liu et al. [48] propose BFSDA, a blockchain-based secure double auction protocol. Using secret sharing and Pedersen commitments, they construct an interactive round-based protocol where participants submit the quantity they want to buy/sell at the current price, resembling a Walrasian mechanism. These offers are then secret shared amongst all participants, and each participant verifies that their received share is correct. Each participant then aggregates the received shares and broadcasts this to reconstruct overall demand and supply at the current price. Using the Pedersen commitment, the participants check the consistency of shares. The participants find the market-clearing price where demand equals supply in a logarithmic number of rounds. However, their scheme requires the interactivity of all participants to resolve the protocol. Furthermore, participants cannot bid on all prices at once but only

on the current price in each round, hence requiring interactivity in all rounds.

Wang et al. [49] propose a double auction protocol based on McAfee's double auction protocol [53]. To find the auction result in the McAfee double auction, the auctioneer sorts the buy and sell orders and finds the least profitable transaction k and the first $k-1$ buy and sell orders get to trade their items. Firstly, buyers and sellers generate a distributed ElGamal key, and each trader thus owns a share of the key. Bidders and sellers then create a vector of ElGamal encryptions using the distributed key, where each index represents a price. The vector for buyers consists of encryptions of two until the price they want to buy at, and then encryptions of one for ascending prices. Sellers create the same vector but in reverse, as their offer represents the lowest price at which they want to sell. Traders prove the correctness of the vector format in zero-knowledge. The auctioneer then receives these offers and multiplies the ciphertexts together for each price. The auctioneer then finds the number of buyers and sellers for each possible price by asking the traders to perform a distributed ElGamal decryption. Then, the auctioneer uses the McAfee mechanism to find the least profitable transaction and the market-clearing price. The auctioneer computes the winners by asking the traders to decrypt the offers at the market-clearing price. By using distributed ElGamal decryption, the scheme requires the participation of all traders, which is expensive in terms of communicational costs and makes the scheme less flexible. Furthermore, it only allows an offer at one price.

3.2.3. Auctioneer: Single Server

Liu et al. [50] propose a periodic double auction running inside an enclave provided by Intel SGX. Intel SGX provides a trusted execution environment. Outside parties can verify that the correct code is running using a process called *remote attestation*. Furthermore, secret keys are stored in *enclaves* and are only accessible to certain processes. In theory, the properties provided by Intel SGX ensure that code runs in an environment where an attacker cannot extract intermediate results. Liu et al. use this to create an auctioneer that computes the result inside an isolated process. In this way, even the auctioneer only learns the final result. Their scheme makes sure that early termination causes economic penalties by using smart contracts and deposits. However, since the introduction of Intel SGX, researchers have studied various attacks requiring patches [54], and Intel has recently stopped integrating Intel SGX on desktop consumer hardware [55]. Hence, in practice, a malicious auctioneer may break this scheme and view data inside the enclave by using one of the attacks.

Xu et al. [51] propose a scheme using homomorphic encryption and sorting networks to implement a privacy-preserving Mcafee mechanism [53]. The authors note the implications of the lack of confidentiality in a double auction protocol which malicious traders may abuse. Traders submit their offers as encrypted binary vectors of their offer, using the Goldwasser-Micali homomorphic encryption scheme. The encryption key used is that of a third agent. The auctioneer and the third agent cooperate in comparing the offers using a 1-bit circuit using `XOR` and `AND` gates. The sorting circuit uses the 1-bit comparator to sort and find the market-clearing price. The authors outline different methods to sort the offers and compare them. The scheme preserves offer confidentiality but only allows bidding on one item, and participants cannot verify the result. Lastly, the construction to achieve an `AND` gate requires interaction between the auctioneer and the third agent, and thus the overall complexity of the comparison computation increases.

Sarenche et al. [21] propose a scheme for smart grid electricity trading. By considering the low power available in smart-grid devices, they design a scheme with a low-communication overhead and low round complexity. Furthermore, their scheme is flexible to use in any procedure for double auctions. Traders get tokens from a trusted control center for different categories of double auctions of traders with similar demand and supply. Traders then place bids/asks in a category from which they hold a token. In the first phase, traders send a commitment to the auctioneer, after which they get a pseudonym. Using this pseudonym, traders generate their actual offers, encrypt them with the public key of the auctioneer, and submit them to the auctioneer, who decrypts them. The auctioneer then uses the decrypted offers to compute the market-clearing price. However, while encrypting the offer, the offer gets revealed to all users participating in the same auction category (albeit anonymous). Furthermore, the auctioneer gets to compute the auction result in plaintext. Thus the confidentiality of bids against the auctioneer is not preserved. In the scheme, all participants must send messages, even if they are not participating in the auction, creating additional overhead.

Galal et al. [52] propose a verifiable periodic double auction for dark pools. Their scheme considers the malicious behavior of traders by using zero-knowledge-proofs. Participants commit to their bid and later send the same offer encrypted to the auctioneer. At the same time, participants prove that their encryption hides the same information as the commitment, which a smart contract verifies. The auctioneer decrypts all offers, sorts the bids and asks by price, and finds the price that trades the highest volume. Lastly, the auctioneer creates an order of the volume traded at the market-clearing price, inserts it into the bids and asks, and proves the correctness of that position to a smart contract. The authors show that the protocol could make sense for small-scale dark pools. While verification plays a large role, there is no confidentiality of offers towards the auctioneer. Hence, the auctioneer learns all offers in plaintext. Especially in dark pools, one may argue that confidentiality plays a vital role. Furthermore, the scheme does not allow participants to have multiple offers simultaneously. Lastly, their scheme is expensive regarding gas/transaction fees due to the expensive operations needed for malicious behavior detection.

Researchers have also studied privacy-preserving spectrum auctions [56–59], where parts of the electromagnetic spectrum are traded. While solutions use a double auction mechanism, they often have additional parameters to consider. For example, spectrum auctions have a matching problem where participants may conflict due to the simultaneous use of frequency bands. Furthermore, location plays an important role in spectrum auctions, which is not important in double auctions. Due to these constraints, such schemes are not easily adaptable; thus, we will not further study them.

3.3. Electricity Trading

Since electricity trading is the primary use of our periodic double auction, we also investigate privacy-preserving electricity trading solutions. Researchers proposed several solutions here. However, many solutions focus on peer-to-peer protocols, requiring interactivity between participants, such as smart homes. Interactivity between participants may be problematic due to the low-level hardware often present. Furthermore, most schemes provide electricity trading on a smaller scale, e.g., microgrids, communities localized to a part of the city, or individual cities. The double auction protocol for electricity exchanges that we study is better suited on a national or international scale that aggregates all offers to gather liquidity of electricity to trade. Hence, the scenarios are different.

Firstly, some of the works previously mentioned indeed provide solutions for electricity trading using periodic double auctions [21, 46]. However, as mentioned the scheme by Abidin et al. [46] does not provide verification and the scheme by Sarenche et al. [21] does not provide confidentiality of offers.

Xie et al. [60] propose PEM, a Private Energy Market in which producers and consumers trade their electricity in a privacy-preserving way. The authors note that individual energy demands and supplies are sensitive information and should thus be protected. Their solution uses homomorphic encryption and garbled circuits to realize a comparison. Initially, each participant creates Paillier key pairs. Then, buyers and sellers group into a buyers and sellers coalition. Buyers aggregate their demand in a round fashion by encrypting their demand with the public key of a chosen seller r_1, adding a random nonce to their demand, adding it to the existing demand, and sending it to the next buyer. The last buyer sends it to the sellers, who add their randomness and send it to r_1 at the end, which decrypts the ciphertext. The same procedure is repeated for sellers, using the public key of a chosen buyer r_2. Each previous entity reuses the randomness they used in the first round. r_1 and r_2 then work together on a secure comparison protocol using garbled circuits. The market-clearing price is then determined, and each trader gets their respective share. However, their scheme assumes semi-honest adversaries, and thus the real-world applicability of the protocol is questionable, as traders can cheat. Furthermore, interactivity between all participants is required to resolve the protocol. In particular, the scheme randomly chooses two participants to conduct the comparisons, which requires more work.

Gaybullaev et al. [61] propose a novel way to represent an encrypted quantity that efficiently computes the inner product to evaluate an integer comparison using functional encryption. Functional encryption is similar to homomorphic encryption, except that the result of the operation is directly available in plaintext. The authors propose a smart contract that receives the encoded and encrypted buy and sell offers and maintains a heap structure to store the highest offer for buying and the lowest offer for selling in the root of the heap. When a new offer comes in, the offer is inserted into the heap structure

by standard heap operations using the proposed comparisons. After restoring the heap order, the smart contract compares the two root elements to find a match. A match occurs if the buying price is higher than the selling price. However, due to relying on a smart contract to construct the heaps, the operations are expensive in transaction costs. Furthermore, functional encryption and smart contracts may not provide confidentiality. All parameters of the smart contract and any transactions that the smart contracts processes are visible to the public. Hence, an adversary may brute force the price in the root by evaluating the functional encryption at ascending prices until the result changes.

Lastly, multiple solutions rely on the blockchain to create an energy trading market [62, 63]. While providing some tools to protect privacy, such as pseudonymity or digital signatures, they fail to consider the confidentiality of quantities. Furthermore, the trust is transferred from a central authority to the nodes of the underlying blockchain and thus goes in a similar direction as the works using multiple parties where a majority is honest. The difference is that general blockchains require gas fees, which participants must pay. Lastly, they often disregard the possibility of verifying the result.

3.4. Summary

Many works are tackling the issue of privacy and verification in double auctions. However, to the best of our knowledge, no periodic double auction scheme with multiple offers takes confidentiality and verification into account without interactivity between participants. At the same time, few works study the double auction with a Walrasian mechanism. We note that some schemes consider privacy and verification in electricity trading. However, no scheme tackles the privacy and verification in (inter)national electricity exchanges on current platforms to the best of our knowledge.

4

PPVDA: Privacy-Preserving Verifiable Double Auctions

The following section introduces PPVDA, a Privacy-Preserving Verifiable Double Auction scheme tailored to electricity trading. PPVDA is suited for double auctions where participants can submit multiple offers at different price points, to have a flexible position, and the auction evaluates periodically. This chapter describes the entities, assumptions, and protocols that compose PPVDA.

4.1. Setting and Assumptions

Before introducing the scheme, we establish the setting and assumptions for our scheme. The setting is the entities involved, their capabilities, and their relationships. Furthermore, we describe the assumptions of the auction and cryptographic assumptions of the scheme.

4.1.1. Entities

Traders

Traders are buyers and sellers who want to trade electricity. Our scheme assumes that malicious buyers and sellers are present in the auction. Thus traders can deviate from the protocol and, e.g., provide negative quantities to buy or sell, which negatively affects the auction procedure. However, the scheme's design will ensure that malicious behavior is detectable.

Auctioneer

The auctioneer is the central entity that receives all the offers from traders and aggregates the offers to compute the overall demand and supply. The auction platforms introduced earlier are examples of auctioneers. In our scheme, we assume that the auctioneer may also be malicious but is economically rational due to the concerns outlined in Chapter 1. Economic rationale means that the auctioneer will try to maximize their profit from the auction and not, for example, block the procedures completely or randomly block traders from trading without knowing their identity.

Third Agent

The third agent is an entity that helps the auctioneer in calculations and is assumed to be honest-but-curious. The third agent may be a government body that ensures the auction runs correctly. Indeed, in electricity trading, the European Union has an interest that the auction runs correctly [11]. However, one crucial feature of the third agent is that they are only involved in a subset of the calculations.

Bulletin board

For the construction of our scheme in the malicious case, a public bulletin board is present to keep track of offers submitted by traders. The public bulletin board is accessible to the general public and thus presents the information all entities may learn. However, the bulletin board is append-only, meaning data is not deletable. The bulletin board should not be maintained by the auctioneer themselves.

4.1.2. Relationships

We assume two client-server models. Traders interact with the auctioneer and the bulletin board. Thus traders do not know each other and have no direct links. The auctioneer interacts with the traders, the third agent, and the bulletin board. Finally, the third agent interacts with the auctioneer and the bulletin board.

We assume authenticated secure channels exist between any two entities when communicating. We also assume that each entity has a public-private key pair, and any entity a communicating with an entity b has the authenticated public key of b.

4.1.3. Assumptions

Auction

Given electricity trading as a setting, we assume the following parameters for the auction. Participants submit (multiple) quantity-price pairs on prices between €-500 and €3,000, in steps of €0.01, as done at current leading electricity trading platforms [13, 14]. Hence, there are 350,000 different prices to submit quantities on by multiplying each possible price by 100 to move the decimal point to the right. The quantities one can submit are, in practice, unlimited. However, we assume that the aggregated quantity does not exceed a 32-bit number; thus, the maximum demand/supply at any price point is $2^{32} - 1$. This limit is changeable for other auctions but is crucial for later purposes. Traders submit quantities in steps of 0.1 MWh. Hence, we move the decimal point by interpreting an encrypted quantity as ten times the actual quantity. Traders submit up to 200 quantity-price pairs per auction. Again, this limit is changeable.

One crucial assumption is that of the economic rationality of buyers and sellers. We assume that buyers and sellers follow the laws of supply and demand. Hence, buyers want to buy more at low prices and buy less at high prices. Sellers want to sell little at low prices, and more at high prices [5]. Traders will prove this property to ensure at most one intersection point between supply and demand. We also assume that there indeed exists an intersection point between supply and demand, while in reality, it is possible for supply and demand never to intersect [14].

Cryptographic assumption

We also make a few assumptions about cryptographic problems and their hardness.

We assume the hardness of the mathematical problems behind the cryptographic schemes we use. We assume the hardness of the factoring problem, decisional composite residuosity problem, discrete logarithm problem, and decisional Diffie-Hellman problem. Furthermore, we act in the random oracle model, assuming the existence of cryptographic hash functions to act as random oracles.

4.1.4. Properties

While there are many properties that privacy-preserving auctions can satisfy, research suggests that satisfying them all is not possible [64]. Indeed, intuition would agree with this. For example, designing a fully anonymous and traceable auction is not trivial, especially in the malicious case. Hence, it is not desirable to satisfy all properties but focus on a few. PPVDA aims to satisfy the following properties introduced in Chapter 2:

- Unforgeability
- Pseudonimity
- Traceability
- Public verifiability
- Non-repudiation
- Confidentiality

4.2. Design

In the following section, we present the design of PPVDA. Overall, PPVDA consists of five steps:

1. Traders register with the third agent and auctioneer
2. Traders submit their offers to the auctioneer (and the bulletin board)
3. After the auction ends, the auctioneer aggregates all buy and sell offers into demand and supply curves.
4. The auctioneer and third agent cooperate to compute the market-clearing price and quantity
5. Winners are identified and trade their items

Traders have to register before submitting their offers. However, traders do not have to re-register for each subsequent auction once registered. Thus steps 2-5 are relevant for each auction period. Given these five steps, we treat them individually to design PPVDA. We first introduce the honest-but-curious case and extend the construction to the malicious case to streamline the process. Figure 4.1 presents the whole scheme in the malicious case.

Throughout this document, we make use of the definitions defined in Table 4.1.

Symbol	Explanation
$a\|\|b$	Concatenation of a and b
$C(\cdot), C$	Generate Pedersen commitment, Pedersen commitment
$e_a(\cdot), E$	Paillier encryption with key a, Paillier Encryption
$H(\cdot)$	Cryptographic hash function
L	Message space of inputs, in our case 32-bit
(pk_a, sk_a)	Public-private keypair of entity a
$RP(x), RP$	Range proof using Bulletproofs on number x, Range proof
$Sig_a^{Alg}(\cdot)$	Cryptographic Signature using private-key a with Alg either RSA or Ed (Ed25519)
(tpn, tpk)	Pseudonym and public key associated to pseudonym
TS	Timestamp or random number
T	Token consisting of pseudonym and signatures on pseudonym
u	ECEG message space, i.e. 256-bit
$Ver_{pk_a}(\cdot)$	Verify RSA or Ed25519 signature
$x \leftarrow\$$	Sample x from a distribution
ZKP	Zero-knowledge proof of consistency
$\mathbb{Z}_n$	Integers up to n
$\mathbb{Z}_n^*$	Integers up to n excluding 0
$[\cdot]$	Paillier ciphertext
$[[\cdot]]$	Elliptic Curve-ElGamal ciphertext
λ	Security parameter of Cut and Choose

Table 4.1: Table of symbols and explanations

4.2.1. Honest-but-curious Case

In the honest-but-curious case, all entities act according to the protocol but are curious about any information they can learn. The general idea is to use homomorphic encryption so the auctioneer can aggregate offers into supply and demand curves. After the auction ends, the auctioneer runs a secure comparison protocol with the help of the third agent. This protocol finds the market-clearing price using a binary search, using the laws of supply and demand.

Registration

In the first step of the protocol, traders need to register to receive authorization to participate in subsequent auctions. Since pseudonymity is required, traders cannot just register with the auctioneer with their ID. At the same time, it shall be possible to trace the identity of traders. VPPDA uses both the auctioneer and the third agent to realize this behavior. Traders first present their ID, create pseudonyms,

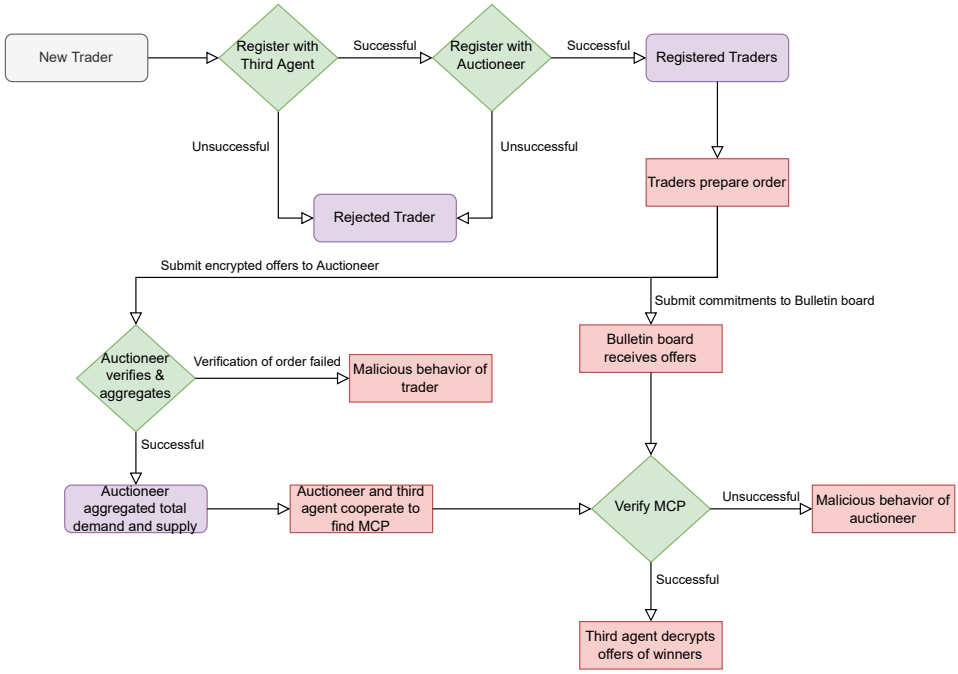
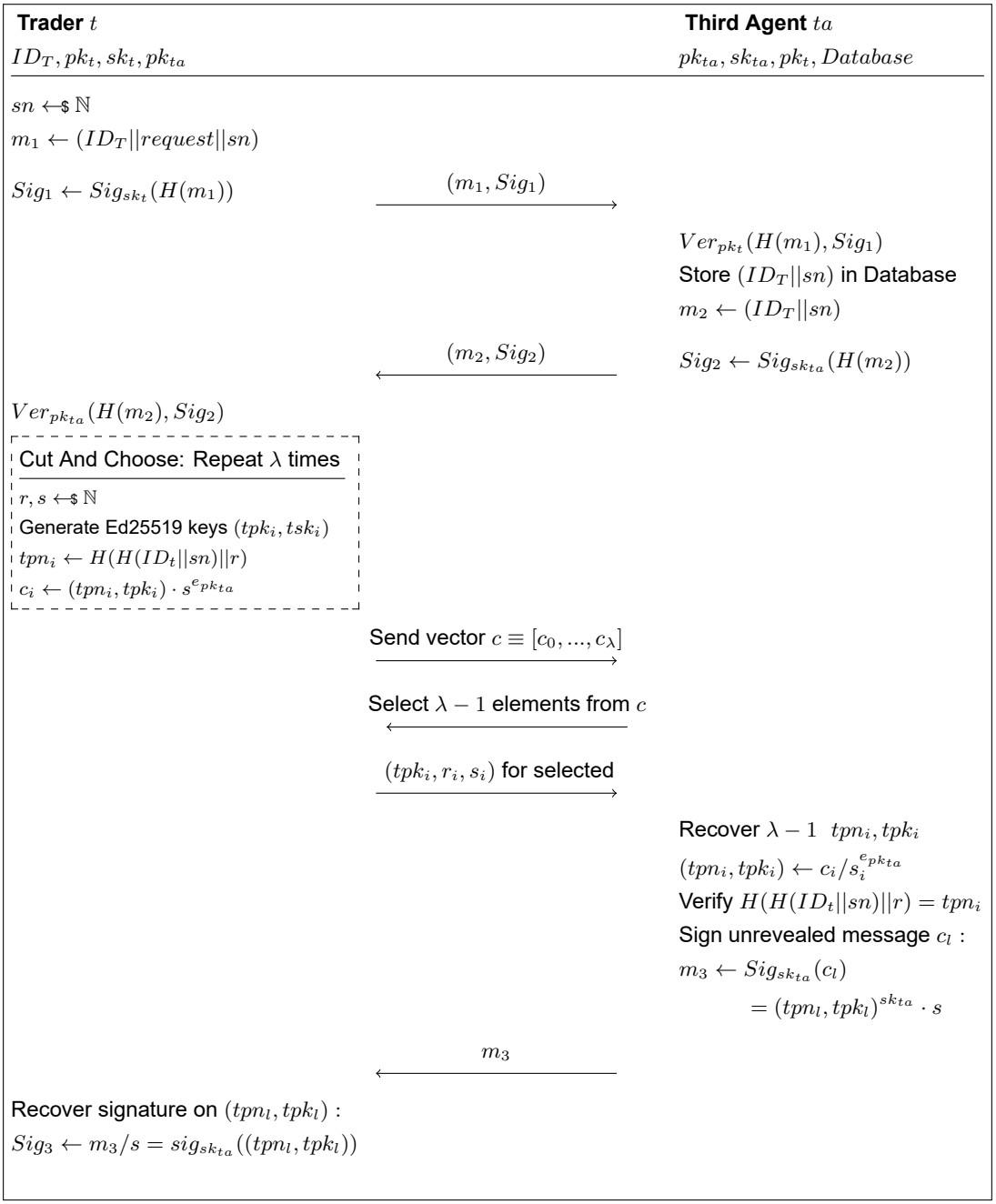


Figure 4.1: Flow diagram of proposed double auction scheme

and get a blind signature from the third agent on their pseudonym. The third agent records the information on the ID of traders and stores it in case of disputes. With the signature on the pseudonym, traders register with the auctioneer to get a token to trade in upcoming auctions. Wang et al. [65] propose a registration procedure with these properties. Their scheme is already secure in the malicious case. Hence, we present the construction for the malicious case directly and show how to adapt them to the semi-honest case. We introduce their two sub-protocols for clarity.

Protocol 4.1 presents the first step, in which traders communicate with the third agent to receive a signature on their pseudonym. Traders have an identification ID to present, a public-private key pair for themselves (pk_t, sk_t), and the public key of the third agent pk_{ta}. The third agent also has a public-private key pair (pk_{ta}, sk_{ta}) and the authenticated public key of the trader pk_t. We note that the signature scheme at the traders' side can be any signature scheme suitable for digital signatures, such as `Ed25519`, while the scheme at the third agent's side needs to provide blind signatures such as `RSA`. Initially, traders create a random number sn to generate a new request to send to the third agent and append that to their real-life ID. Traders sign this with their private key and present it to the third agent. The third agent verifies the signature and acknowledges the request by returning a signature on the sent message. The third agent stores $(ID||sn)$ used in the requests for future purposes, and traders store the signature received on their request.

Given this interaction, traders want a blind signature on a pseudonym they create. In Protocol 4.1 we present the more complicated malicious case, making use of *Cut and Choose*. In the malicious case, the concern is that traders generate pseudonyms of the wrong format. However, in the honest-but-curious case, traders will always produce pseudonyms of the correct format; hence, the third agent will believe that the format is correct and blindly sign it. Traders generate two random numbers r, s. They generate a new `Ed25519` keypair (tpk_i, tsk_i) and a pseudonym tpn_i by computing $H(H(ID_t||sn)||r_i)$. Due to the use of cryptographic hash functions, the output will look completely random. Traders encrypt s with the public of the third agent, multiply that with (tpn_i, tpk_i), and send this to the third agent. The pseudonym-public key pair (tpn_i, tpk_i) looks random now due to using a random blinding factor. The third agent signs the blinded term with their private key, which cancels out parts of the blinding term (see Protocol 2.1). Traders recover a signature on the pseudonym-public key pair by dividing by the original blinding term s. Due to blind signatures, the third agent has no information on the signed pseudonym and cannot relate it to the previously presented ID.



Protocol 4.1: Registration by Wang et al. [65]: Interaction between Trader and Third Agent

After traders receive a blind signature on their pseudonym, they interact with the auctioneer to get a token to participate in auctions. The auctioneer has their own public-private key pair (pk_a, sk_a), the public key of the third agent pk_{ta}, and the public key of the trader pk_t. The trader will send $H(ID_t||sn)$, a signature on $H(ID_t||sn)$ of the third agent which they previously stored, the randomness r_l associated with their pseudonym, their pseudonym-public key pair (tpn_l, tpk_l) and the third agent's signature on the pseudonym-public key pair, to the auctioneer. The auctioneer then checks that they have not seen this hash before and verify the signatures of the third agent. If all the signatures are correct, the auctioneer verifies that the pseudonym is correctly formatted. If the verification is correct, the auctioneer also signs the pseudonym-public key pair and sends the signature to the trader, who assembles their token T.

Trader t		**Auctioneer** a
$ID_T, sn, r_l, (tpn_l, tpk_l), pk_a$ $Sig_1 \leftarrow Sig_{sk_{ta}}(tpn_l, tpk_l)$		$pk_a, sk_a, pk_t, Database$
$m_1 \leftarrow H(ID_T\|\|sn)$ $\|\|Sig_{ta}(H(ID_T\|\|sn))$ $\|\|r_l\|\|(tpn_l, tpk_l)$ $\|\|Sig_{sk_{ta}}(tpn_l, tpk_l)$		
	$m_1 \longrightarrow$	
		Check $H(ID_T\|\|sn) \notin Database$ Verify signatures on $H(ID_T\|\|sn)$ and (tpn_l, tpk_l) Verify $tpn_l = H(H(ID_T\|\|sn)\|\|r_l)$ $Sig_2 \leftarrow Sig_{sk_a}((tpn_l, tpk_l))$
	$\longleftarrow Sig_2$	
$Ver_{pk_a}(Sig_2)$ $T \leftarrow \{(tpn_l, tpk_l), Sig_1, Sig_2\}$		

Protocol 4.2: Registration by Wang et al. [65]: Interaction between Trader and Auctioneer

Each trader must run this procedure with the third agent and the auctioneer to register. Traders assemble their token T by their pseudonym-public key pair (tpn_l, tpk_l), the third agent's signature, and the auctioneer's signature on it. Traders will use the token T each time they submit offers and use the corresponding private key to sign offers.

Bidding

An auction period is open in a certain time interval in which traders submit their offers to the auctioneer. Generally, a raw offer has the form: (q_i, p_i), a quantity-price pair, where the offer quantity is q_i, and the corresponding price is p_i. A trader groups many orders to submit demand or supply at ascending prices. Traders indicate whether they want to buy or sell with an indication ("B" or "S"). Furthermore, traders use the token T generated and include it in their offer so the auctioneer checks that the trader has registered. Traders use their tokens to sign their quantity-price pairs with their private `Ed25519` key corresponding to their pseudonym. The quantities a trader wants to trade are sensitive information. Thus traders encrypt their quantities with the public key of the third agent using the Paillier encryption scheme and send the encryptions to the auctioneer. Using the third agent's public key ensures that the auctioneer does not learn the quantities. Hence, when buyers want to submit quantity-price pairs to the auctioneer, they submit the message:

$$order = \{B, T, [(e_{pk_{ta}}(q_0), p_0), (e_{pk_{ta}}(q_1), p_1), ..., (e_{pk_{ta}}(q_n), p_n)], Sig_T^{Ed}(offers)\}. \tag{4.1}$$

Similarly, when sellers want to submit quantity-price pairs, they submit the message:

$$order = \{S, T, [(e_{pk_{ta}}(q_0), p_0), (e_{pk_{ta}}(q_1), p_1), ..., (e_{pk_{ta}}(q_n), p_n)], Sig_T^{Ed}(offers)\}. \tag{4.2}$$

We note that the prices are ascending and that traders should make an offer at price 0 and price 350,000 for simple aggregation in the implementation. For simplicity, we refer to the indication of buying or selling as `order_type`, the token as `T`, the vector of quantity-price pairs as `offers`, and the signature on the quantity-price pairs as `Sig_offers`. The auctioneer receives these orders and processes the quantity-price pairs to aggregate.

Aggregation
When an auctioneer receives an offer, they process it to update current demand and supply at all prices. The auctioneer computes the updated demand and supply curve by adding the new offer to the received aggregated offers. Since the auctioneer does not have the private key corresponding to the public key used in the encryption, the auctioneer must add in the ciphertext domain using homomorphic addition. The auctioneer finds the correct quantity in the trader's order for each possible price and adds it to the aggregated demand and supply. If a buyer has not submitted an offer on a specific price, the auctioneer takes the encrypted quantity for the next highest price where the buyer did submit a quantity. Indeed, by inspection of the demand in Figure 2.2, we see that the quantity at, e.g., price 11 corresponds to the same quantity as price 15. Similarly, if a seller has not made an ask for a specific price, the auctioneer takes the encrypted quantity for the next lowest price. Again, we confirm this by looking at the supply curve.

Algorithm 1 Aggregating offers

```
1:  demand, supply ← [0; 350,000], [0; 350,000]   ▷ Aggregated demand, supply. Initialize with encryption of 0
2:  procedure AGGREGATE(order)                    ▷ incoming order
3:      Initialize pointer to point to first quantity-price pair (offer) in order
4:      if buy order then
5:          for i in 0..350,000 do
6:              if price of current offer is equal to i then
7:                  Add quantity of current offer to demand at price i
8:                  Increase pointer to point to next quantity-price pair
9:                  Next iteration of loop
10:             end if
11:             Add quantity of current offer to demand at price i
12:         end for
13:     else if sell order then
14:         for i in 0..350,000 do
15:             if price of next offer is equal to i then
16:                 Increase pointer to point to next quantity-price pair
17:             end if
18:             Add quantity of current offer to supply at price i
19:         end for
20:     end if
21: end procedure
```

In Algorithm 1 we present the procedure for the auctioneer to aggregate offers, assuming homomorphic addition and quantity-price pairs for ascending prices. For an incoming order $order$, the auctioneer first determines whether the order is a buy or sell order to add quantities for all 350,000 prices. We loop through all prices ascendingly and keep a pointer pointing to a quantity-price pair called *current offer*. If the order is a buy order and the pointer points to a quantity-price pair equal to the current price, we take the quantity for that price and increase the pointer for the next iterations to take the quantity for the next highest price. In the case of selling, we increase the pointer if the current price hits the next quantity-price pair. This exactly corresponds to the graphical representation from Figure 2.2. The auctioneer receives offers as long as the auction window is open. Hence, this procedure runs repetitively until the auction closes for incoming orders.

Compute the Market-clearing-price
Given supply and demand at different price points, the auctioneer aims to find the price at which supply and demand intersect, as this is the price at which traders trade the most. The auctioneer uses a binary search to find the highest price for which demand exceeds supply. The following price is the lowest price for which supply is greater than or equal to demand. The intersection and thus the market-clearing price is one of these two points, depending on which one maximizes the quantity to trade. The auctioneer uses a binary search as this only requires $\lceil \log(350,000) \rceil = 19$ comparisons.

Since the traders encrypted the quantities with the third agent's public key, the auctioneer cannot compute comparisons of quantities themselves. We use a secure comparison protocol to compare values between the auctioneer and the third agent to guide the binary search. This problem is a variant of the millionaire's problem [41]. The auctioneer holds two encrypted values a, b encrypted with the public-private key-pair of the third agent and would like to learn the result of $a < b$ and nothing else. That is, the only thing that any entity should learn is $a < b$. The auctioneer cooperates with the third agent to compute this result, using supply and demand as their respective a, b. The comparison result should be public to run the binary search. The protocol we use in the semi-honest case is by Nateghizad et al. [66]. We present the protocol for secure comparison in Protocol 4.3.

The basic procedure aims to compute the most significant bit of $a - b$ for two encrypted values $[a], [b]$, by calculating:

$$z_l \leftarrow 2^{-L} \cdot (2^L + a - b - ((2^L + a - b) \pmod{2^L})), \tag{4.3}$$

where L is the binary length of the two numbers a, b to compare. In essence, the subtraction ensures that only the first L bits are set (from behind), and the multiplication shifts the result to the correct position. Since the two numbers $[a], [b]$ are Paillier ciphertexts, it is easily possible to add and subtract them. However, the difficulty lies in the computation of the modular term. One way to compute the modular reduction is to decrypt the term and compute it in the plaintext domain. Hence, the task is to compute $(2^L + a - b \pmod{2^L})$ together with the third agent. We use implicit encryptions using $[\cdot]$ and $[[\cdot]]$ notation for Paillier and Elliptic-curve-ElGamal, respectively, with the public key of the third agent. The auctioneer computes $[2^L + a - b + r]$, to create the relationship of $a - b$, but hiding the exact difference with the randomness r. The third agent decrypts this, computes the modular term, encrypts it, and sends it back. The auctioneer now computes

$$[d' - r'] = [2^L + a - b + r \pmod{2^L} - r \pmod{2^L}], \tag{4.4}$$

which is exactly the term we need. However, the calculation may result in a negative value if $r' > d'$. Thus, the auctioneer and third agent must compare their inputs d' and r'. If $r' > d'$, the auctioneer needs to add 2^L to turn the negative number into a positibe number. Hence, the auctioneer and the third agent run another comparison protocol where they compare r' at the auctioneer's side and d' at the third agent's side, but this time with unencrypted inputs to compute $r' > d'$. The goal is to create encryptions for every bit position, where the encryptions are zero only at the first position where $d'_i < r'_i$, which signalizes that r' is greater than d'. This step is the computation of $[[t_i]], [[v_i]]$ in Protocol 4.3 and is an improvement by Nateghizad et al. to the equation introduced by Erkin et al. [67]:

$$c_i \leftarrow s + d'_i - r'_i + 3 \sum_{j=i+1}^{L-1} d'_j \oplus r'_j. \tag{4.5}$$

The auctioneer sends the ciphertexts to the third agent, which performs an ECEG Zero-check and sets the bit δ accordingly. The bit δ is again encrypted with Paillier and corrected at the auctioneer's side to compute the final encrypted result. Lastly, the auctioneer sends the final result to the third agent for decryption.

We note that the original protocol from Nateghizad et al. [66] uses DGK [68]. DGK is an additively homomorphic scheme that is more efficient than Paillier due to a smaller ciphertext space. However, DGK suffers from the same problem as ECEG of needing to know the message range roughly. We use ECEG due to faster key generation and general speed of elliptic curve operations. Nateghizad et al. also propose a faster method to compute z_l, which we also use but is equivalent to the one presented in Equation 4.3.

Auctioneer A		**Third Agent** ta
$[a] =$ aggregated supply $[b] =$ aggregated demand $L =$ Length binary representation		
$r \leftarrow\!\!\$\ \mathbb{Z}_{\kappa+L}$ $[d] \leftarrow [2^L + a - b + r] = [2^L] \cdot [a] \cdot [b]^{-1} \cdot [r]$		
	$[d] \longrightarrow$	
		$d' \leftarrow d \pmod{2^L}, \hat{d} \leftarrow d/2^L$
	$\longleftarrow [d'], [\hat{d}]$	
$r' \leftarrow r \pmod{2^L}, \hat{r} \leftarrow r/2^L$		
		$[[t_i]] \leftarrow [[d'_i + \Sigma_{j=i+1}^{L-1} d'_j \cdot 2^j]]$
	$\longleftarrow [[t]] \equiv [[t_0, ..., t_{L-1}]]$	
$s \leftarrow\!\!\$\ \{-1, 1\}$ $[[v_i]] \leftarrow [[s - r'_i - \Sigma_{j=i+1}^{L-1} r'_j \cdot 2^j]]$ $[[c_i]] \leftarrow [[v_i]] \cdot [[t_i]]$ $h_i \leftarrow\!\!\$\ \mathbb{Z}_u^*$ $[[e_i]] \leftarrow [[c_i \cdot h_i]] = [[c_i]]^{h_i}$		
Scramble order of $[[e_i]]$	$[[e_i]] \longrightarrow$	
		Zero-Check all $[[e_i]]$ If none are zero, $\delta' \leftarrow 0$ else, $\delta' \leftarrow 1$
	$\longleftarrow [\delta']$	
$[\delta] \leftarrow [\delta']$ if $s = 1$ $[\delta] \leftarrow [1] \cdot [\delta']^{-1}$ if $s = -1$		
$[z_l] \leftarrow [\hat{d} - \hat{r} - \delta] = [\hat{d}] \cdot [\hat{r}]^{-1} \cdot [\delta]^{-1}$		
	$[z_l] \longrightarrow$	
		Decrypt $[z_l]$ and release result

Protocol 4.3: Secure comparison protocol by Nateghizad et al. [66]. Determines whether $a < b$

By using Protocol 4.3 as a subroutine for the binary search, the auctioneer finds the highest price, for which demand $>$ supply. The following price is the lowest price for which demand $\leq$ supply. To find the exact market-clearing price, we note that we aim to maximize the quantity traded, which is one of these two prices. Hence, the auctioneer sends the aggregated demand and supply at both prices to the third agent. The third agent computes the minimum between supply and demand at both prices, as those will be the eventual quantities traded at the two prices. Then, the third agent compares the two minimums and takes the price with the higher minimum to be the market-clearing price. The quantity at that price is the quantity that traders will trade.

Identify Winners
Given the market-clearing price, the auctioneer forwards the offers made at this price to the third agent. The third agent decrypts all offers and declares them publicly. The pseudonym of the trader is bound to the offer; hence, the public knows the pseudonyms of winners. Traders now step forward and engage in the transfer of electricity.

4.2.2. Malicious Case

Given the construction for the semi-honest case, we extend the construction to address malicious behavior. We assume that traders and the auctioneer may act differently at any step than the protocol in the semi-honest case. PPVDA uses a mix of zero-knowledge proofs and Pedersen commitments to detect malicious behavior.

The malicious model aims to cover similar attacks and concerns as presented in Chapter 1, and our construction will make these attacks detectable. In particular, we consider the following attacks in the auction:

- Traders providing negative quantities or not monotonically increasing/decreasing quantities. This affects the correctness of the binary search for the market-clearing price and can make it fail
- The auctioneer providing wrong offers as part of the aggregations after learning a favorable market-clearing price. This directly manipulates the correctness of the market-clearing price
- The auctioneer inserting bids and asks after the auction has closed, thus manipulating the correctness of the auction.

It is important to note that we are addressing malicious but economic rational behavior. We assume that an auctioneer may be interested in explicitly excluding selected participants from the auction but will not randomly exclude traders of whom they do not know the real-life identity.

Registration
The registration procedure introduced in Protocols 4.1 and 4.2 is already secure in the case of malicious traders. Using Cut and Choose, the registration procedure is secure against a malicious trader up to a security parameter λ. In particular, instead of creating a single pseudonym, traders create λ pseudonyms and λ `Ed25519` key-pairs. Traders will blind all the pseudonym- public key pairs and send them to the third agent. However, since traders may want to cheat and create pseudonyms of the wrong format, the third agent will ask traders to uncover $\lambda - 1$ of the pseudonym-public key pairs to verify their structure. Traders provide the blinding terms s_i and randomness r_i used for each requested pair, which the third agent uses to unblind and verify the structure. If all of them are of the correct structure, the third agent believes the remaining pseudonym-public key pair is also correct and blindly signs it. The security here relies on the fact that the third agent gets to choose which pseudonym-public key pairs to reveal.

The interaction between traders and the auctioneer stays unchanged.

Bidding
Submitting offers in the semi-honest model consists of an indication of whether the trader is buying or selling, a token, several encrypted quantities, and a signature on the offers. To secure this construction in the malicious model, traders must prove a few fundamental properties of the encrypted quantities. The concern is that malicious traders submit negative or non-monotonically increasing or decreasing quantities. These may manipulate the market-clearing price and make the search for the market-clearing price fail. While we assume economically rational actors, traders may be interested in

creating multiple intersection points between demand and supply to cause confusion requiring more comparisons than usual. The construction will make sure this is not possible.

Traders continue to submit encrypted quantities to the auctioneer who uses them for aggregation. However, traders will also prove to the auctioneer that their encrypted quantities are inside a certain range and are ascending or descending. We accomplish this by using Pedersen commitments which traders publish onto the public bulletin board. Traders provide Pedersen commitments on plain quantities, on which they prove the properties using range proofs such as Bulletproofs [37]. The submitted commitments have a similar structure as `offers`. However, instead of encrypted quantities, traders provide Pedersen commitments. Assuming that traders also prove that a ciphertext and a commitment hide the same message, traders prove properties on the ciphertext by proving it on the commitment and establishing the equivalence of the messages hidden in the two. That is, traders want to prove a property on the ciphertext, establish an equivalence between the encryption and commitment, and prove the property on the commitment to automatically prove the same property on the encryption. Indeed, this technique is used in previous works [69].

Firstly, we introduce the construction to prove the equivalence between a Paillier ciphertext E and a Pedersen commitment C to the auctioneer. We utilize a Sigma protocol by Jurik [69] and refer to the proof in the original work. We present the Zero-knowledge proof of consistency between a ciphertext and a commitment in Protocol 4.4.

Trader		**Auctioneer**
$(g_1, n), (g_2, h, q), m, r, s$		$(g_1, n), (g_2, h)$
$E \leftarrow g_1^m r^n$		E, C
$C \leftarrow g_2^m h^s$		
$\hat{m} \leftarrow\!\!\$\ \mathbb{Z}_L$		
$\hat{r} \leftarrow\!\!\$\ \mathbb{Z}_n$		
$\hat{s} \leftarrow\!\!\$\ \mathbb{Z}_q$		
$a \leftarrow g_1^{\hat{m}} \hat{r}^n$		
$b \leftarrow g_2^{\hat{m}} h^{\hat{s}}$		
	$a, b \longrightarrow$	
	$\longleftarrow e$	
$x \leftarrow \hat{m} + m \cdot e$		
$y \leftarrow \hat{r} \cdot r^e$		
$z \leftarrow \hat{s} + s \cdot e$		
	$x, y, z \longrightarrow$	
		Accept iff
		$g_1^x y^n = a \cdot E^e$
		$g_2^x h^z = b \cdot C^e$

Protocol 4.4: Zero-knowledge proof of consistency between encryption and commitment by Jurik [69]

In Protocol 4.4, traders want to convince the auctioneer that the provided Paillier encryption E and Pedersen commitment C hide the same message (quantity) m that only the trader should know. The public parameters for Paillier (g_1, n) and Pedersen commitments (g_2, q, h) are assumed to be known to the public, and the auctioneer does not hold the secret key for the corresponding Paillier parameters. The trader generates new parameters $\hat{m}, \hat{r}, \hat{s}$ and provides them as part of the ZKP commitment step, using them to compute a new Paillier ciphertext and a new Pedersen commitment. In the interactive version, traders send these to the auctioneer. The auctioneer provides a random challenge to the traders, who use the challenge for a computation using the homomorphic property. The relationship between the encryption and the commitment is proven based on the common parameter x, which ensures that the encryption and commitment hide the same message, assuming that finding the same commitment with a different message and randomness is hard (discrete logarithm problem).

Protocol 4.4 is an honest-verifier zero-knowledge proof. Using the Fiat-Shamir Heuristic, we transform the interactive nature of Protocol 4.4 into a non-interactive zero-knowledge proof. The interactive challenge step is replaced by computing the challenge as the hash of inputs, i.e., $H(E, C, a, b)$, where H is a cryptographic hash function. Using the Fiat-Shamir heuristic thus allows traders to prove the equivalence to the auctioneer without any interactivity required. Traders will submit Paillier encryptions to the auctioneer and send Pedersen commitments to the public bulletin board. Traders also create the proof in Protocol 4.4 for each pair of encryptions and commitments and send the proofs to the auctioneer. The auctioneer verifies the proofs using the verification step of the protocol. Thus, using this proof, traders prove that the encryptions and the commitments hide the same message. The auctioneer should check these consistency proofs as soon as they receive an order. The offer should be excluded from the auction if the proof does not hold.

Given the consistency between the encryption and the commitment, a trader now proves properties on their commitments using range proofs. The auction setting requires that the quantities fit inside an unsigned 32-bit integer. Each trader generates a range proof of their quantity for each quantity-price pair, proving that the quantity hidden in their commitment is an unsigned 32-bit integer. Proving the monotonicity is also possible with range proofs using a similar idea as presented in Galal et al. [52]. Buyers want to prove descending quantities, and sellers want to prove ascending quantities. Buyers reverse their quantity-price pairs and prove that the difference between successive quantities, i.e., $q_{i+1} - q_i$ is also an unsigned 32-bit integer. Thus, buyers create range proofs for the value hidden by the difference of successive elements, using the parameters of the elements. Sellers do the same without reversing their quantity-price pairs to prove ascending quantities.

To protect against the offer being repeated by a malicious trader, traders append timestamps or a random number TS to the end of their `offers`, which will also get signed. The auctioneer and bulletin board will also check that they have not seen this offer yet, based on the value at the end.

Overall, the construction looks as follows: Traders send the same messages as in the honest-but-curious case to the auctioneer but append zero-knowledge proofs of consistency ZKP_i for each quantity-price pair and add randomness TS to the `offers`. Traders create the same kind of message using commitments instead of encryptions and publish that to the public bulletin board. Traders append the Bulletproofs range proofs RP_i on their quantities instead of ZKP_i, allowing any verifier to verify that their quantities are inside the range and ascending or descending.

An offer to the auctioneer now consists of:

$$order = \{B/S, T, [(e_{pk_{ta}}(q_i), p_i), ..., TS], Sig_T^{Ed}(offers), ZKP_i\}. \tag{4.6}$$

An offer to the public bulletin board consists of:

$$order = \{B/S, T, [(C(q_i), p_i), ..., TS], Sig_T^{Ed}(offers), RP_i\}. \tag{4.7}$$

Aggreation
The construction to aggregate does not change compared to the honest-but-curious model. The auctioneer does not reveal any computations on the aggregates until the next step when the auction closes. Hence, we cannot check if the auctioneer aggregated correctly since no information was released.

Compute the market-clearing price
The auctioneer runs a binary search with the help of the third agent to find the market-clearing price. We present a construction to verify the market-clearing price using economic properties.

The auctioneer runs a binary search to find the highest price for which demand $>$ supply. The following price is the lowest price for which demand $\leq$ supply. Since there is only one intersection, these are the only two prices at which this switching between demand and supply occurs, and hence the region of prices is correct. The market-clearing price is one of the two, depending on which maximizes the quantity to trade. Hence, the third agent verifies the correct market-clearing price by using the economic structure present in the supply and demand at the two possible market-clearing prices. The third agent will receive the aggregates for demand and supply from the auctioneer at the two prices and decrypt them. Then, the third agent verifies the correctness by comparing the supply and demand at both prices.

However, the auctioneer could create offers to steer the market-clearing price towards a more lucrative price while presenting an economically correct market-clearing price. Here, we use the public bulletin board. The bulletin board contains commitments with timestamps on when they were submitted. Hence, it is possible to view which offers were made in time before the auction closed. For each offer at the two prices, the auctioneer will send the encryptions and consistency proofs they previously received to the third agent. The third agent obtains all commitments at the two prices from the bulletin board and uses the consistency proofs to verify the consistencies between the encryptions and commitments. Hence, the third agent verifies that all commitments have a zero-knowledge proof of consistency and corresponding encryption. If all offers on the public bulletin board at that price have corresponding encryption, the third agent knows that the auctioneer did not cheat. Then the third agent runs the verification mentioned above.

Identify Winners
The auctioneer already forwarded the offers at the correct market-clearing price to the third agent for verification. Hence, the third agent decrypts the offers made at the market-clearing price and knows the exact quantities that are bought and sold at the market-clearing price. The pseudonyms of winning traders are attached to the offers, and the third agent thus reveals the pseudonyms. Traders that have won now step forward and engage in the transfer of electricity. In the case of traders not stepping forward, the third agent works together with the auctioneer to reveal a trader's identity, based on the traceability property [65].

4.2.3. Withdrawing Orders

We also propose extending the scheme to cover functional properties such as withdrawing orders. Traders may withdraw orders at any period before the end of the auction. When traders want to withdraw an order, they send a cancellation offer with a particular flag set. The cancellation offer looks exactly like the original offers, except for the special flag, and the trader sends it to the auctioneer and the public bulletin board. Hence, the auctioneer receives the encryptions, and the bulletin board receives the commitments. The auctioneer verifies that they received the offer before, verifies the consistency proof in the cancellation offer, and homomorphically *subtracts* the encrypted quantities from the aggregates. The bulletin board now contains both the original offer and the cancellation offer. Since the Pedersen commitment is also additively homomorphic, the offer also cancels if the cancellation offer is subtracted from the original offer. However, it leaves a trace of both offers on the public bulletin board. To ensure other entities except the original sender do not cancel offers, we require the trader to include the cancellation flag in the signatures. Given a canceled order at the public bulletin board, the third agent can safely disregard the verification of that offer when verifying consistency proofs.

5 Evaluation

In this chapter, we evaluate the design of PPVDA by arguing how the scheme meets its security properties and analyzing the computational and communicational complexity. We implement and run our auction protocol to gain insights into the runtimes of specific operations.

5.1. Theoretical Analysis

This section evaluates the scheme's properties and the computational and communicational complexities. We first provide sketches on how the properties are met and then analyze the computational and communicational complexity of several defined operations.

5.1.1. Security

Unforgeability

Traders should not be able to impersonate other traders when they submit offers to the auctioneer. The scheme uses `Ed25519` signatures to ensure that only the owner of the private key corresponding to the pseudonym can sign the offers. The key pair is generated in the registration procedure using a Cut and Choose protocol. It ensures that only the trader knows the private key associated with the public key and thus is the only entity that can sign offers. A malicious entity who recovered the private key associated with the pseudonym will be able to break this property. However, recovering the private key through the public key is intractable in polynomial time due to the discrete logarithm assumption. Hence, the scheme meets the unforgeability property, assuming the hardness of the discrete logarithm problem and assuming a trader keeps their private key secure.

Malicious entities may also attempt a replay attack by recording an offer made by a trader and replaying the same offer with the same signature. While the signature will be valid, the randomness used at the end of the `offers` is the same. Hence, the auctioneer and bulletin board have seen the same message twice and will be able to detect that the second message is a replay of the first and subsequently ignore the message. To ensure that traders can submit the same offer in two different auction periods, we introduce an "auction-Id" that identifies the auction. Prepending the "auction-Id" to the `offers` would ensure that an offer is bound to the specific auction. Adding the Id prevents a malicious trader from replaying the same offer in a different auction period.

Pseudonimity

The use of blind signatures in the registration process enables the pseudonymity of traders [65]. The registration procedure in the first step ensures there is no link between the real ID of a trader and the pseudonym created since the third agent does not see the pseudonym they sign. When traders interact with the auctioneer in the registration step, they only present their pseudonym and a hash of their ID appended with a random number. Hence, the auctioneer does not have enough information to recover a trader's identity. Only the auctioneer and third agent can together uncover the identity of traders. We assume no collusion between the auctioneer and the third agent to preserve the pseudonymity.

When submitting offers, traders only reveal their pseudonyms and a signature using their pseudonym-key pair and thus remain pseudonymous. We also note that traders may have multiple pseudonyms active simultaneously to use at varying times. Furthermore, we recommend generating a new pseudonym after a trader has won in an auction to preserve pseudonymity since the identity may be revealed. Traders may even choose to generate new pseudonyms for each auction period to leak as little information about the offers previously submitted.

Traceability
The scheme requires traceability to identify the real identity of malicious traders, which the registration procedure provides. The auctioneer and third agent work together to reveal the identity of a trader and their pseudonym [65]. The auctioneer searches their database for an entry $H(ID_T||sn)$ corresponding to the pseudonym in question with randomness r_l. The auctioneer sends this information to the third agent, who searches their database for the ID corresponding to this entry.

A malicious trader may try to cover their identity in the registration process by creating a pseudonym in the wrong format. The best strategy here is for malicious traders to insert one pseudonym in the wrong format in a position where they think the third agent will not ask for the revealing factors. The probability of the malicious pseudonym not being chosen is $1/\lambda$, as traders reveal $\lambda - 1$ pseudonyms and keep one pseudonym secret. Hence, choosing a large λ is important e.g. $\lambda = 2000$. In practice, even if there is a probability of cheating with pseudonyms, the low probability of staying undetected and the fear of being excluded from future auctions may scare traders not to act maliciously in the registration, especially since they present their real identification.

In practice, the coalition of auctioneer and third agent can thus recover the real identity of a trader. An auctioneer may abuse this and pretend to receive incorrect signatures or zero-knowledge proofs to find a trader's identity or ban them from the auction. However, we believe that this is not a significant concern. Firstly, we argue that a rational economic auctioneer will not abuse traceability. The auctioneer has little information on the real identity of traders (due to confidentiality of quantities and pseudonymity). In a real-world system, the auctioneer should earn money for every trade (as the service should not be free). Hence, the auctioneer is not interested in randomly leaving out participants but only specific traders, as seen in Chapter 1 due to economic gains for every trade. At the same time, the auctioneer should only get to report traders acting maliciously before the comparisons to ensure that an auctioneer does not report on offers when computing a lucrative market-clearing price.

We also note that the auctioneer cannot simply "incriminate" another trader: Suppose an auctioneer wants to incriminate a trader A with the wrongdoing of a trader B. The auctioneer easily recovers $H(ID_A||sn_A), H(ID_B||sn_B)$ for both traders. However, the auctioneer should also reveal the parameters r_A, r_B used by traders in their pseudonym to create $pn_A = H(H(ID_A||sn_A)||r_A), pn_B = H(H(ID_B||sn_B)||r_B)$. For both traders, this is easy as traders send their parameter r in the registration procedure. However, the auctioneer would have to find an r' that generates for the ID of A, the pseudonym of B. That is, the auctioneer would have to find an r' such that $pn_B = H(H(ID_A||sn_A)||r')$. However, this would break the second-pre-image resistance of cryptographic hash functions and is thus a contradiction.

Public Verifiability
Public verification allows any entity to verify that the auction result is correct. We achieve this by using a mix of cryptography and economic properties. The binary search finds the highest price for which demand $>$ supply, but the correct market-clearing price may be this or the next price. The auctioneer requires the assistance of the third agent to decrypt the aggregated quantities at the two prices to compute the correct market-clearing price, depending on which maximizes the quantity to trade. Firstly, the auctioneer releases the encryptions at the two prices and the consistency proofs to the public. The public and the third agent verify the consistency between the encryptions and the commitments on the bulletin board using the consistency proofs. This consistency verification ensures that the auctioneer included the correct offers.

The third agent then decrypts the aggregated supply and demand offers at the two prices to verify that the two prices make sense. At the correct intersection, the demand at the lower price will be lower or equal to the supply at the next price. Hence, we verify whether the region is correct by examining the

supply and demand at the two prices and comparing them. This check ensures that the price is correct given the encrypted quantities. We note that the decryption is needed for verifiability in general, as the market-clearing price is the intersection of two line segments, which requires two points for each line. Without decryption, the auctioneer could present a different price for which the economic property does not hold. Hence, we must decrypt at the two prices to verify the correct result and compute which price maximizes the quantity to trade. We note that Bogetoft et al. require a similar procedure in their scheme [12].

Non-repudiation
When traders submit offers, they append a signature of their offer signed with the private key associated with their pseudonym. The randomness traders append at the end of the quantity-price pairs ensures that a malicious entity cannot replay an offer with the same randomness. Due to the properties of Ed25519 signatures, traders cannot deny having made an offer as long as their private key is still private, under the assumption of the hardness of the discrete logarithm problem. Since the public bulletin board is *append-only*, traders cannot remove information which adds another layer of non-repudiation.

Confidentiality
Confidentiality protects the sensitive information about quantities that traders fear an auctioneer could abuse. Traders submit Pedersen commitments to a public bulletin board and Paillier encrypted quantities with the public key of the third agent to the auctioneer. Furthermore, traders provide zero-knowledge-proofs of consistency and range proofs. We note that the Pedersen commitments are information-theoretic hiding and thus leak no information as long as the parameters do not leak. Traders encrypt their quantities with the third agent's public key and then send them only to the auctioneer. Thus the auctioneer cannot read the quantities under the assumption of the composite residuosity problem. The zero-knowledge proof and bulletproof construction inherit their properties from their original works [37, 69]. Hence, we argue that the encryptions, commitments, and proofs do not leak information by themselves.

One opportunity for a malicious auctioneer to leak information is by abusing secure comparisons. The auctioneer can present arbitrary values a, b to learn more about a trader's offer in the comparisons. However, the auctioneer must run the secure comparisons to learn the actual market-clearing price. Furthermore, the number of comparisons the third agent runs is limited to $\lceil \log(350,000) \rceil = 19$ comparisons since that is the needed number to learn the market-clearing price. Hence, while the auctioneer has the opportunity to learn comparison results and present arbitrary values, the auctioneer also has to learn the correct market-clearing price to present to the third agent to not risk detection of malicious behavior. Since the correct market-clearing price has two decimal places, the auctioneer will not be able to guess the correct MCP without some correct comparisons easily. However, an auctioneer will likely be able to guess the rough region of the correct MCP through previous results. Hence, the auctioneer may require fewer comparisons due to the smaller region, resulting in a few "free" comparisons which the auctioneer may use to learn values. Thus indeed, complete confidentiality is not achieved. However, due to pseudonymity, the auctioneer cannot identify the trader behind an offer and thus only make comparisons with the offer of a random trader. Hence, while an auctioneer has opportunities to leak the plaintext of offers, its use may be limited.

Verifying the market-clearing price requires revealing the offers at two prices, one of which will be the eventual market-clearing price. Hence, we leak offers at a second price, also conflicting with the confidentiality property we aimed to achieve, as the third agent can decrypt the offers made at the two prices. While this is the case, the number of offers that additionally leak is likely low as traders would have had to specifically submit a quantity-price pair at this price to get a different quantity than the quantity that would be decrypted at the market-clearing price anyway.

We note that the structure of the offers leaks information about the prices of interest for a trader. For example, submitting an offer at a price already leaks the interest in that price, and an auctioneer may be able to approximate the quantity. Furthermore, continuously submitting offers at the same prices also leaks information. In this design, a trader can include "dummy" offers that do not change the orders but generalize offers prices. For example, a buyer may be interested in prices 10 and 20. The buyer could insert the same quantity for price 20 into price 15. Traders can also submit quantities of 0 for higher prices, which due to the semantic security of Paillier, are indistinguishable from other

ciphertexts. Actually, given a curve, traders can submit all quantity-price pairs going through the curve as part of the offers while preserving the meaning of the curve. The auctioneer now does not know which prices exactly are of interest. The hiding of interested prices can be generalized by submitting offers on all prices, which would likely be too much computational overhead. A suitable balance between submitting offers and hiding prices of interest is thus desirable. Another option would be for all traders to follow a similar distribution for their quantity-price pairs. Using the same distribution would prevent the auctioneer from learning about a trader's behavior by looking at the prices they are interested in and protecting confidentiality better. We leave this as future work.

5.1.2. Computational and Communicational Complexity

We analyze the scheme's computational and communicational complexities regarding the primitive operations we define. We present these operations in Table 5.1, which is largely based on Table 4.1. In the case of factoring-based solutions such as Paillier or RSA, we calculate the approximate size with a modulus of 3072 bits. In RSA, this results in a 3072 bit ciphertext, and for Paillier, this results in a $2 \times 3072 = 6144$ bit ciphertext. For Bulletproofs, we refer to the original paper for a 32-bit range proof [37]. For Pedersen commitments and Elliptic-Curve operations, we refer to documentation regarding `Curve25519` [30, 70, 71]. For the zero-knowledge proof of consistency ZKP, we note that the encryptions and parameters $\hat{m}, \hat{r}$ are the largest factors. We estimate a size of $(3 \times 2 \times 3072)/8000 \approx 2.304$ KB.

Operation	Explanation	Approx. Size (KB)
C	Pedersen Commitment	0.032
$D : Paillier$	Decryption Paillier	–
$E : ECEG$	Elliptic-Curve-ElGamal encryption	0.128
$E : Paillier$	Paillier encryption	0.768
$E : RSA$	RSA encryption	0.384
$H(\cdot)$	Cryptographic hash function such as SHA-256	0.032
$Hom.Add$	Add homomorphically	–
$Hom.Exp$	Homomorphic Exponentation	–
$KG : Ed$	Key-generation for Ed25519	0.032
$Mod.Inv$	Modular Inverse	–
RP	Generate Range proof using bulletproofs	0.610
$Sig : RSA$	Generate RSA signature	0.384
$Sig : Ed$	Generate Ed25519 signature	0.064
Ver	Verify RSA or Ed25519 signature	–
$VerRP$	Verify Range proof	–
$VerZKP$	Verify Zero-knowledge proof of consistency	–
ZC	ECEG Zero-check	–
ZKP	Generate Zero-knowledge proof of consistency	2.304

Table 5.1: Operations executed by entities and sizes of outputs

Computational

We examine the operations executed at each step of the protocol for each entity. We focus on cryptographic operations such as generating Paillier encryptions or zero-knowledge proofs of consistency.

Traders Traders have three interactions with other entities where they carry out computations. When traders communicate with the third agent, they mainly generate pseudonyms and encryptions as blinding factors for their pseudonyms. Hence, the running time is mainly influenced by the factor λ. When traders register with the auctioneer, they only send some previously recorded information to the auctioneer and then verify the signature from the auctioneer. The number of offers a trader generates and submits decides the number of encryptions, range proofs, commitments, and consistency proofs necessary. In particular, for every offer, a trader generates a Paillier ciphertext, a Pedersen commitment, a consistency proof, and a single range proof. When traders generate n offers, they additionally need to generate $n-1$ range proofs proving ascending or descending quantities, hence overall needing $2n-1$ range proofs. The order is then signed. Table 5.2 presents an overview of the operations.

Operation	C	E	KG	RP	Sig	Ver	ZKP
Registration λ							
Trader $\leftrightarrow$ Third Agent	–	$RSA:\lambda$	$Ed:\lambda$	–	$Ed:1$	$RSA:2$	–
Trader $\leftrightarrow$ Auctioneer	–	–	–	–	–	$Ed:1$	–
Submit n **offers**							
Generate offers	n	$Paillier:n$	–	$2n-1$	$Ed:1$	–	n

Table 5.2: Overview of computational complexity for a single trader

Auctioneer The auctioneer carries out three essential steps in the auction. The auctioneer verifies proofs, aggregates the offers, and participates in the secure comparison with the third agent. The verification of offers is simply verifying the tokens and their signatures (an RSA signature and an Ed25519 signature), the signature on the order, the range proofs, and consistency proofs to ensure that the order is valid. Each verification process runs independently of the other and is parallelizable. Once the offers are verified, they are used to generate an aggregate at each price. For this, the auctioneer needs n homomorphic additions.

The auctioneer then cooperates with the third agent to run the secure comparison. Here several steps are needed, for which we refer back to Protocol 4.3. In particular, the auctioneer creates $L = 32$ ECEG encryptions, homomorphically adds to them, and multiplicatively blinds them. A constant number of homomorphic additions and modular inversions are needed to complete the protocol. Table 5.3 presents an overview.

Operation	E	$Hom.Add$	$Hom.Exp$	$Mod.Inv$	Ver	$VerRP$	$VerZKP$
Receive n **offers**							
Verify offers of single trader	–	–	–	–	$Ed:2$ $RSA:1$	$2n-1$	n
Aggregate n **orders**							
At single price	–	n	–	–	–	–	–
Comparison							
Secure comparison	$ECEG:L$ $Paillier:3$	$ECEG:L$ $Paillier:6$	$ECEG:L$	$Paillier:4$	–	–	–

Table 5.3: Overview of computational complexity for the auctioneer

Third Agent The third agent has three main phases. They participate in the secure comparison, help find the final result, and declare the final winners. For the secure comparison, the third agent does similar work as the auctioneer with the ECEG encryptions, except that the third agent also runs an ECEG Zero-check. For the final result, the third agent verifies the consistency proofs, verifies the signatures posted on the bulletin board, decrypts aggregates made at the two possible market-clearing prices, and declares them publicly. This requires 4 decryptions of aggregated quantities. The signatures on the pseudonyms and offer are also verified, as well as the consistency and range proofs. The third agent then decides the market-clearing price by selecting the price that maximizes quantity. The third agent decrypts the buy and sell orders at the market-clearing price to declare the auction winners, requiring $2m$ decryptions. Table 5.4 provides an overview of the operations.

Public verification is very similar to the verification step of the third agent without the decryption step.

Operation	D	E	$Hom.Add$	Ver	$VerRP$	$VerZKP$	ZC
Comparison							
Secure comparison	$Paillier:2$	$ECEG:L$ $Paillier:3$	–	–	–	–	L
Find correct MCP and verify							
Verify m offers included	$Paillier:4$	–	$Paillier:4m$	$Ed:2\times 4m$ $RSA:4m$	$4m$	$4m$	–
Final result							
Identify m winners	$Paillier:2m$	–	–	–	–	–	–

Table 5.4: Overview of computational complexity for the third agent

Summary We summarize the computational complexity for the auction without registration as follows: Let n denote the number of offers per trader and m the number of traders, assuming all traders submit the same number of offers. The complexity of traders is linear in the number of offers. The complexity for the auctioneer depends on the number of traders and offers per trader. The number of comparisons is a constant term. The complexity for the Third Agent is linear in the number of traders submitting orders. Table 5.5 summarizes the computational complexities.

Entity	Computational Complexity
Trader	$O(n)$
Auctioneer	$O(mn)$
Third Agent	$O(m)$

Table 5.5: Overview of computational complexities

Communicational
The communicational complexity is the size of the messages participants need to send to one another. We use the above-presented information to study how much data is sent between the entities. In particular, we focus on the main steps of the scheme, composed of the registration procedure, preparing and sending offers, and the secure comparison. We estimate the size of the messages to be 0.064 KB if not otherwise specified. Table 5.6 provides an overview of the computational complexities for a strong security parameter λ and the maximum number of offers n.

Operation	Approx. Size (KB)	Approx. Size (KB) $\lambda = 2000, n = 200$
Trader		
Registration Trader $\leftrightarrow$ Third Agent	$\|ID\| + 0.544\lambda - 0.096$	$\|ID\| + 1087.904$
Registration Trader $\leftrightarrow$ Auctioneer	0.928	0.928
Send n offers	$4.322n - 0.610$	863.79
Auctioneer		
Registration Trader	0.064	0.064
Secure comparison	$1.54 + 0.128L$	5.636
Third Agent		
Registration Trader	$0.064\lambda + 0.696$	128.696
Secure comparison	$2.31 + 0.128L$	6.406

Table 5.6: Overview of computational complexity for the auctioneer

In the registration procedure, traders interact with the third agent and the auctioneer. In the interaction with the third agent, traders send 1 signature and message, λ RSA ciphertexts and $\lambda - 1$ public keys (and parameters r, s). The size of the ID is $|ID|$. This results in a size of $|ID| + 0.064 + 0.384\lambda + (0.032 + 0.064 + 0.064)(\lambda - 1) = ID + 0.544\lambda - 0.096$ KB. For the registration of traders with the auctioneer, traders send two hashes, two RSA signatures, one Ed25519 public key, and a random number, resulting in a size of $2(0.032) + 2(0.384) + 0.032 + 0.064 = 0.928$ KB. To send n offers, we need n Paillier ciphertexts, n commitments, n consistency proofs and $2n - 1$ bulletproofs, resulting in: $0.768n + 0.032n + 2.304n + 0.610(2n - 1) = 4.324n - 0.610$ KB.

For the registration of traders at the auctioneer's side, the auctioneer only has to send an Ed25519 signature, which is 0.064KB in size. For the registration of traders at the third agent's side, the third agent sends an RSA signature with a message, selects $\lambda - 1$ indices, and sends a blind RSA signature. This results in $0.380 + 0.064 + 0.064(\lambda - 1) + 0.380 = 0.064\lambda + 0.696$.

For the secure comparison, the auctioneer sends two Paillier ciphertexts and L ECEG ciphertexts to the third agent, resulting in $2 \times 0.77 + 0.128L = 1.54 + 0.128L$ KB. The third agent sends three Paillier ciphertexts and L ECEG ciphertexts, resulting in a size of $3 \times 0.77 + 0.128L = 2.31 + 0.128L$ KB.

Overall, communication size rarely exceeds a Megabyte of data, except in the registration procedure. However, we note that traders can reuse pseudonyms, and thus, the registration procedure does not have to run before each auction period. Hence, from a communicational complexity perspective, we believe our scheme is usable in the real world.

5.2. Practical Analysis

While we analyzed the scheme theoretically, it is perhaps more interesting to examine how the scheme performs in terms of runtime in realistic settings. We implement our design and use artificial test cases to assess our scheme.

5.2.1. Implementation Setup

We implement a proof-of-concept in Rust, where each entity gets a single thread and communicates over channels with no delay. Most notably, we use the Scicrypt [72] library for the encryption schemes Curve25519-dalek-ng [71] for Elliptic curve operations, Ed25519-dalek [70] for `Ed25519` digital signatures and Bulletproofs [73] for the Bulletproofs and Pedersen commitments. We simplify our implementation in various parts. For example, our implementation does not open for 24 hours but receives offers and closes after all traders submit their offers to provide insights into the operations. Furthermore, the proof of concept does not use optimizations for performance. For example, it would be more efficient to use a binary search with multiple pointers, and Paillier data packaging [66] generate a single bulletproof to prove multiple numbers to be in a range [37], or batch verify bulletproofs.

We use security parameters equivalent to AES-128 security to ensure that encryptions are secure. Hence, we use a Paillier and RSA modulus of 3072 bits. Ed25519 and ECEG also provide AES-128 security by default. We evaluate our implementation of the double auction scheme on a machine with an Intel Core i7-7700HQ CPU and 16GB RAM. For each step of the scheme, we present the evaluation times. The largest setting we tested consisted of 200 traders, each submitting 200 offers. Since we evaluate on a single machine, the runtime increases linearly for the number of traders generating offers; hence, we did not test settings with more traders.

5.2.2. Runtime

Preparation

While we assume that public-private key pairs are distributed amongst the participants before the protocol, their generation takes time. Especially factoring-based schemes such as RSA or Paillier take time to generate a key with 128-bit security due to needing to find large prime numbers. Table 5.7 presents key generation time for the different encryption/signature algorithms used in PPVDA. We run the key generation algorithms 100 times and take the mean of the results. We see that the factoring-based solutions require roughly 30 seconds to generate a key, while the elliptic curve-based solutions take 0.05 ms for the same security level. However, the standard deviation indicates significant variability in the key generation for factoring-based solutions, as these depend on finding large primes randomly. The Elliptic curve key generations are particularly fast as only random points need to be generated. We used these results to improve the running time of Protocol 4.1. In the Cut and Choose step, traders generate λ keys, of which one will be used only for signing. To make this procedure as fast as possible, we use Ed25519 for signing purposes, as the key generation is a lot faster than, for example, RSA at the same security level.

Entity	Key	Time
Trader	Ed25519	0.05 ± 0.02 ms
Auctioneer	Ed25519	0.05 ± 0.02 ms
Third Agent	RSA	29.93 ± 22.62 s
	Paillier	31.00 ± 21.35 s
	ECEG	0.05 ± 0.02 ms

Table 5.7: Mean key generation time for keys required by entities with 128-bit security

Registration

Traders register with the third agent and the auctioneer in a two-step registration procedure. We measure the time for an individual trader, the third agent, and the auctioneer to run the procedures for different security parameters λ over 100 traders registering for the auction.

Figure 5.1 presents the runtime for traders, the third agent, and the auctioneer for registration of a single trader for different security parameters λ. In particular, we experiment with $\lambda = 50, 200, 1000, 2000$ where

$\lambda = 2000$ offers the highest security. We observe that the runtime increases for traders for higher values of λ, as traders have to create more pseudonyms and blind more terms. Since the third agent has to verify the pseudonyms, the runtime for the third agent also increases for higher values of λ. The runtime for the auctioneer stays mostly unaffected by higher values of λ. For low values of λ such as $\lambda = 50$, we observe that registration for the trader takes less than 25ms (including interactions with the third agent and auctioneer), 19ms for the third agent, and 0.66ms for the auctioneer. For the highest security level of $\lambda = 2000$, the probability of a malicious trader not being detected is 0.05%. At the same time, the computational runtime is roughly 280 ms for a trader and 112ms for the third agent. We argue that this is fast for a strong security parameter, which would even enable running the registration before every auction period if wanted. The low successful cheating percentage and the fact that a real-life identity is attached will scare traders away from not acting maliciously in the registration procedure, thus achieving traceability.

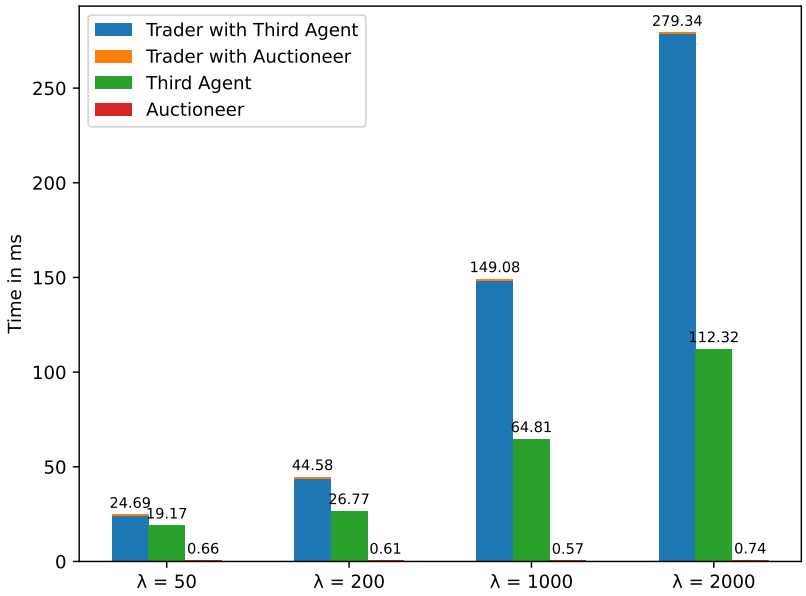


Figure 5.1: Runtime of registration procedure for trader, auctioneer and third agent for increasing security parameters λ

Bidding

Traders prepare and send their offers to the auctioneer and bulletin board in the bidding stage. In Figure 5.2 we present runtimes for generating an increasing number of offers $\{1, 5, 25, 50, 100, 200\}$ across 100 traders. We analyze the running time for running each cryptographic scheme for the traders.

Figure 5.2 presents the required time for generating encryptions, commitments & bulletproofs, consistency proofs, and ascending/descending proofs for an increasing number of offers. Firstly, we observe that the encryptions and consistency proof require the most time, which is not surprising as both require modular exponentiations. We also observe that the bulletproof generation and ascending/descending proof take roughly the same time since they follow the same subroutine with slightly different parameters. Comparing the two groups, the bulletproofs generation is roughly six times faster than the encryptions. It also becomes clear that all operations follow a linear relationship as the number of offers increases.

Looking more closely at a single offer, we observe that single encryptions take 47 ms, the consistency proof 48 ms, and both the bulletproofs and ascending/descending proof 7.1 ms. Hence, most time is spent generating the encryptions, as that is a part of the consistency proofs. For 200 offers, this increases to roughly 9.5 s for the encryptions and consistency proofs and 1 s for the necessary bulletproofs and ascending/descending proofs. We believe the runtime is acceptable for a reasonable number of offers, such as 50 or 100, with which the encryptions take $2 - 4$ s and the range proofs $0.3 - 0.7$ s.

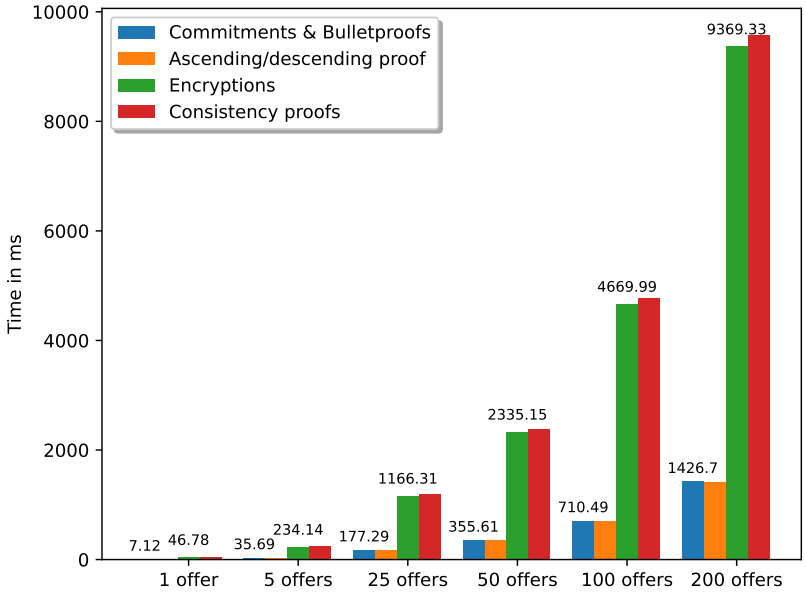


Figure 5.2: Runtime of traders generating offers

Aggregation

The auctioneer receives orders containing multiple proofs and needs to verify these proofs to ensure that the orders are correct. We note that verifying consists of multiple steps. Firstly, the auctioneer verifies the pseudonyms' signatures and the order's signature with the pseudonym. Verifying the token and the signature takes 0.98 ms. Next, the auctioneer verifies the consistency proof using the procedure in Protocol 4.4, which takes 49 ms. The auctioneer is then convinced of the consistency between the encryptions and the commitments on the bulletin board. The auctioneer then verifies the range proofs on the bulletin board. Verifying a single range proof takes 1.04 ms and can be further optimized by batch verification. Hence, while verifying a single quantity-price pair is relatively fast, the times increase as more quantity-price pairs are included in the orders.

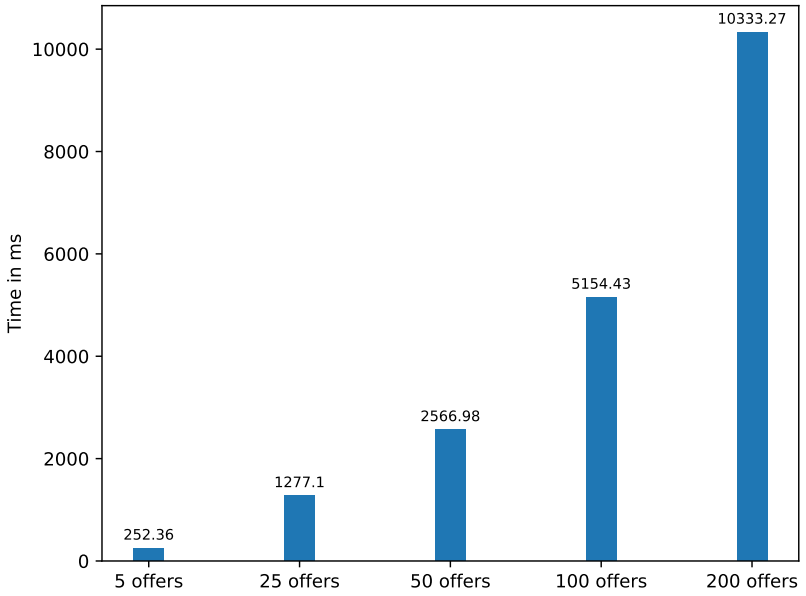


Figure 5.3: Runtime of auctioneer verifying an order of a single trader

Figure 5.3 presents the runtime for verifying a single order from a trader with an increasing number of offers inside the order. In particular, we notice that the time required to verify the order is linear in the number of offers. Verifying an order consisting of 5 offers takes 250 ms while it takes 10 s for 200 offers. Hence, verifying takes a significant amount of time for an increasing number of offers, which is caused by the long time to verify the consistency proof. The verification of the consistency proof contains rather expensive modular exponentiations, which explains the runtime. The auctioneer runs the verification step for each trader submitting an order.

The auctioneer aggregates the received orders for all prices in the aggregation step. In the implementation, we change this procedure to aggregate only for the prices required by the binary search. This optimization decreases memory consumption, as an aggregate is not needed for all $350,000$ prices but only for 19 prices. The number of traders submitting orders determines the runtime, as more traders result in more offers to aggregate at a price. Furthermore, the number of offers per order also determines the runtime. In Figure 5.4 we present the runtime for aggregating offers for $\{1, 5, 25, 50, 100, 200\}$ orders and offers per order for a single price, using homomorphic addition. In particular, we note that aggregating a single order to the current aggregate takes 0.01 ms, and overall aggregating 200 orders with 200 offers takes nearly 90 ms. We observe a linear relationship as the number of orders increases. From this, we argue that aggregating is fast for the auctioneer using homomorphic addition, even for more traders.

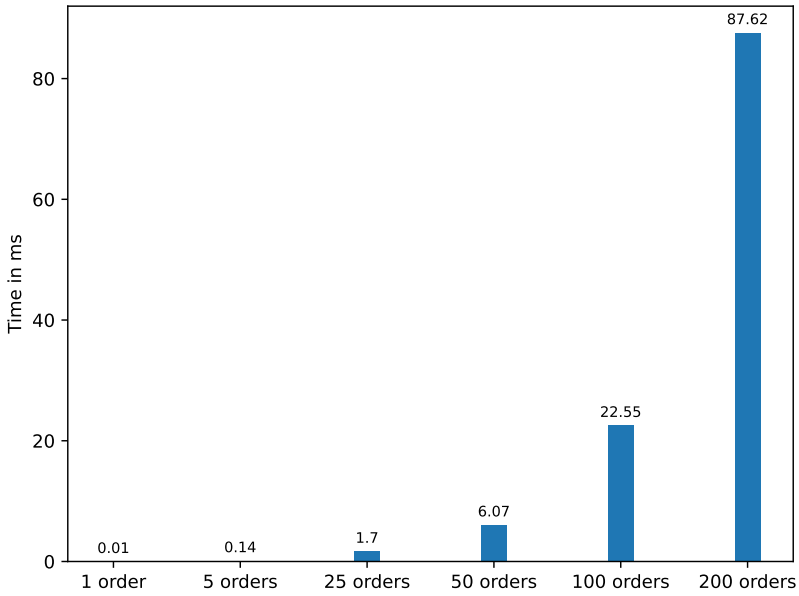


Figure 5.4: Runtime of auctioneer aggregating offers

Compute the market-clearing price

Computing the market-clearing price occurs through a binary search using the secure comparison of Protocol 4.3. We note that the time required for the comparison is not affected by the number of offers submitted or the number of traders submitting offers since they are aggregated in the previous step. Overall, we observe that a single comparison at the auctioneer's side takes 440 ms, while it takes 495 ms for the third agent. This difference is likely due to the more expensive decryptions necessary for decrypting the final result of the comparison. Overall, the binary search takes 11 s.

Identify Winners

The final step of the scheme consists of the auctioneer forwarding the offers at the two possible market-clearing prices to the third agent together with the consistency proofs. The third agent verifies the consistency proofs using the commitments on the bulletin board and verifies the signatures based on the included pseudonyms. In Figure 5.5 we present the runtime of the third agent verifying the consistency of the offers with the bulletin board and decrypting the offers to find the quantity to trade.

We present this for an increasing number of offers. To verify a single offer was included, we verify the consistency proof, which takes 49ms. We observe that the consistency check and decryption take a similar amount of time as the number of orders increases. For 50 orders by traders, this takes $2-3$ s for the consistency check and decryption, while for 200 orders by traders, it takes $9-10$ s.

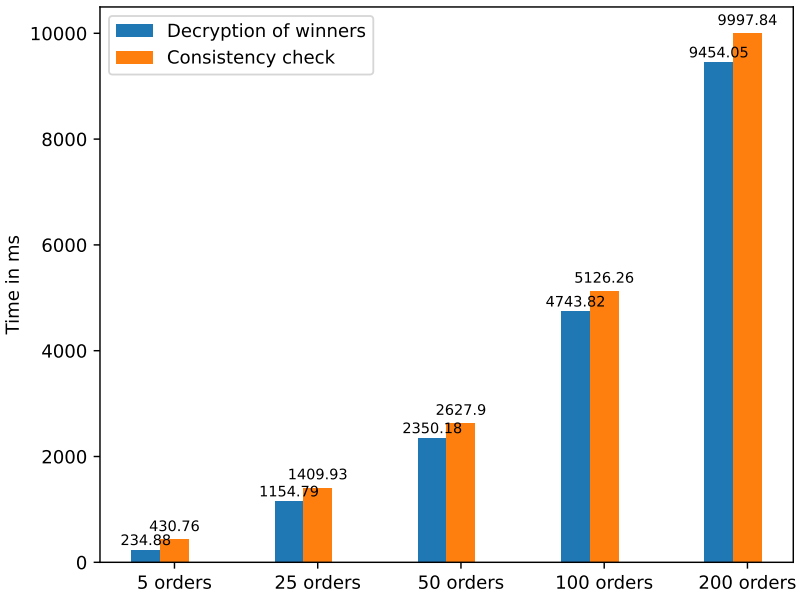


Figure 5.5: Runtime of third agent decrypting winning offers

5.2.3. Summary

In summary, we observe that the scheme is usable for a reasonable number of offers, such as 50 on the trader's side to execute the needed cryptographic operations on consumer hardware. To generate all the needed constructions would roughly take 5 s for 50 offers per trader. The auctioneer verifies the offers and takes 2.5 s to verify 50 offers in an order. Since the auction period is typically open for a 24 hour period, the auctioneer will have enough time to verify the offers as they come in. However, we believe various optimizations in code, such as parallelization, can drastically decrease the time to verify the orders. As the number of traders increases, the third agent must decrypt more offers. The third agent takes roughly 10 s to verify the offers and decrypt them individually to declare the winner for 100 traders submitting orders. Due to the linear relationship, we would expect roughly 100 s to verify and decrypt the orders of 1000 traders, which we believe is a realistic number of traders.

6

Discussion and Future Work

While the general public sees privacy as a desirable property in products and services, it is not the primary goal for many new services [74]. One common belief is that privacy is difficult to do right, makes products or services less appealing, or is slow [74]. Generally, we often trust companies to act ethically and correctly use our data. While laws are in place to prevent unethical or outright malicious behavior, there are examples where companies broke this trust by misusing data, such as Cambridge Analytica [75]. At the same time, we trust companies to execute procedures we cannot monitor ourselves correctly and not abuse their power, such as in auctions or general stock trading. However, one must ask whether we trust these companies too much. For example, allegations of anti-competitive behavior and manipulation are surfacing on popular auction platforms for electricity and advertisements [9, 11], and there are countless more examples of unethical behavior in stock trading such as insider trading [17]. Whether the allegations on the auction platforms are true is unclear, but the possibility of it is not reassuring to participants, and their trust in the system may decrease. It is thus desirable to question the current systems and create solutions that tackle the issue of privacy and correctness at its core.

This work proposes a solution to double auctions' privacy and correctness problem. In this section, we re-evaluate our research questions and discuss our scheme from a critical point of view. We end this work by outlining future research directions.

6.1. Discussion

The main research question in this work was:

How can we construct a privacy-preserving verifiable double auction with confidential and undeniable offers that closely resembles the current auction procedure?

We design a double auction scheme that preserves confidentiality by using homomorphic encryption to hide the quantities a trader wants to buy or sell. Pseudonymity protects the identity of traders to ensure an auctioneer cannot target traders to exclude them from auctions while using digital signatures ensures that offers are undeniable. Overall, we aimed to stick to the current procedure as much as possible and not design a solution that splits the role of the auctioneer over multiple parties. We introduced a new entity, the third agent, who helps in computations that the auctioneer needs to run, such as comparisons. This new entity limits a malicious auctioneer's ability to learn information. We show that our scheme works by implementing a proof-of-concept and evaluating it on our machine. Our implementation shows that the scheme could run on consumer hardware for a reasonable number of offers per trader.

How can an auctioneer aggregate bids and asks and utilize them to compute the market-clearing price securely while preserving confidentiality? We use an additively homomorphic encryption scheme that allows an auctioneer to aggregate offers at the same price to compute aggregated supply and demand. For each price, the auctioneer has aggregated supply and demand and needs to find the intersection between the supply and demand curve. Since the auctioneer does not have the associated secret key, they cannot decrypt individual offers or the aggregate but have to participate in a secure

comparison protocol with the third agent. The use of comparisons ensures that the auctioneer only learns a comparison value that they use to determine the market-clearing price.

How can the public verify the double auction result? Public verifiability ensures that anyone can convince themselves of the correctness of the auction result. We use the information-theoretic property of Pedersen commitments to post commitments onto a publicly accessible bulletin board. The auctioneer presents two prices, of which one is the market-clearing price, and releases encryptions and proofs of consistency. Any entity can use these encryptions and proofs to verify that the encryptions are in sync with the commitments on the bulletin board, ensuring the auctioneer included all offers. The third agent must then ensure that the quantities at the two prices follow the economic property at the correct market-clearing price by decrypting the quantities. Since decryption is needed to identify quantities of winners, we believe this is a suitable and efficient way.

How can malicious behavior of participants be detected? Participants prove vital properties using Cut and Choose, Zero-knowledge proofs, and digital signatures. Cut and Choose ensures that a pseudonym is of the correct format to ensure traceability is possible, and Zero-knowledge proofs ensure, e.g., that a value is inside a range. Since the auctioneer wants to maximize the number of traders and cannot explicitly exclude participants from the auction due to pseudonymity, the auctioneer is interested in truthfully reporting malicious behavior. Hence, when the auctioneer declares that a trader acted maliciously, the auctioneer and the third agent will cooperate in tracing the identity of the trader behind a pseudonym and potentially punish them or exclude them from future auctions. At the same time, the auctioneer should only get to report malicious behavior before the comparisons start to ensure that the auctioneer does not exclude a trader after learning the market-clearing price. The public can check whether all commitments have corresponding encryptions using the commitments on the bulletin board to ensure that the auctioneer did not inject new offers or excluded offers.

6.1.1. Limitations

We note that, naturally, this work has a few limitations. Most notably, traders and the auctioneer cannot be entirely malicious in this scheme. Having two malicious parties that act maliciously simultaneously without rationality is hard to solve. One possible situation would be for trader T to create wrong consistency proofs that the auctioneer A checks. The proof fails, and thus the auctioneer reports to the third agent. However, if asked, T can present the correct proof to the third agent and even pretend that they sent this one to the auctioneer. How does the third agent know who cheated and who did not? We can transfer this to a real-life analogy with breaking the law and there being only one witness and evidence that can easily be faked. One possible solution is to split the trust into multiple auctioneers, which secret sharing solutions typically do. Here, we would trust a majority of the servers to act truthfully. In our case, we explicitly wanted to design a solution with a single auctioneer who receives offers and aggregates, which thus creates these problems. We solve this problem by arguing for the economic rationality of the auctioneer. However, the full extent of this is unclear.

A further point of criticism would be why we assume the existence of a semi-honest third agent. As mentioned before, we aimed to keep the current auction procedure as much as possible intact. While introducing new parties, we wanted to ensure that most operations stayed at the auctioneer's side. Hence, a secret sharing approach was not viable since that would spread the work evenly over the parties. Since the third agent is only needed for decryptions of offers and in the secure comparisons, it may be possible to prove correct decryptions using zero-knowledge proofs. However, we believe a suitable entity for the third agent is a government body; thus, assuming them to be semi-honest is sufficient.

Lastly, we design our scheme with the application of electricity trading in mind. However, electricity trading has many more components. Electricity may be transferred from a region of high supply to a region of low supply, and it is possible for demand and supply to never intersect. Hence, this work only focused on the double auction aspect of electricity trading and not the other parts of the system. Thus, the real-world applicability to real-world electricity trading is unclear. Nevertheless, we believe the scheme is usable for the double auction part of electricity trading and is also interesting for other domains, such as emissions trading or dark pools.

6.2. Future Work

During the design of our scheme, we came across a few interesting lines of work that we describe as possible future work that can be of independent interest.

Malicious comparisons Currently, the scheme relies on economic properties to argue the correctness of the market-clearing price. While this makes the comparisons fast, it allows the auctioneer to carry out comparisons that they should not carry out, which leaks information. We believe the auctioneer can use the consistency proofs between the commitments and the encryptions to prove the correctness of the comparison on the auctioneer's side. Difficulties include the multiplicative blinding factor and the shuffling at the auctioneer's side. However, this would ensure that the comparisons are only to determine the market-clearing price and not to learn offers. The malicious comparisons are also of independent interest to other works using comparisons.

Use Zero-knowledge proofs in registration instead of Cut and Choose The current Cut and Choose method in the first step of the registration procedure only has a security of $1/\lambda$ where λ is the number of repetitions. In contrast, typical zero-knowledge proofs have a security of $2^{-\lambda}$. Hence, using zero-knowledge proofs would provide higher levels of security to prove the structure of the pseudonym while not revealing it. This requires a redesign of the registration procedure, and it would be interesting to see the runtime and interactivity of such a procedure.

Distributed Paillier key over multiple third agents The scheme assumes that a single third agent exists with the secret key for Paillier encryption. Due to this, the encryptions cannot be posted directly to the bulletin board but only sent to the auctioneer. An alternative would be to distribute the key across multiple third agents [76]. This would change the trust model to distribute the trust of the third agent across multiple third agents, each holding a part of the key. The secure comparisons would change, but this would allow releasing encryptions directly onto the public bulletin board. Such a change would get closer to using secret sharing over multiple parties.

Use Polynomials for bidding Instead of defining multiple quantities as an offer and encrypting the quantities, it is possible to define an offer as a polynomial going through several quantity-price points. This change would result in a smooth polynomial computed through Lagrange interpolation at several points. However, it requires finding a monotonically increasing or decreasing curve of a high degree, which is not trivial. The advantage is that a trader always sends encryptions at the same prices to evaluate the polynomial, and aggregating is easy. Finding the market-clearing price is simply solving the equation when the two curves are equal.

Extend the scheme with deposits When traders make an offer, it would be possible for them to pay a deposit to ensure malicious behavior such as providing wrong zero-knowledge proofs becomes economically unviable. Indeed, some works use deposits for these purposes [45, 77]. Traders could pay deposits to a smart contract which would only be returned if traders behaved honestly.

6.3. Concluding Remarks

Double Auctions are, in theory, suitable economic tools to produce a fair and efficient auction result and subsequent allocation. However, when such a system is implemented in the real world, several concerns about the auction's privacy and correctness can arise. This work presents a scheme to design a privacy-preserving and verifiable double auction using cryptography where participants do not have to interact in the whole protocol. We utilize homomorphic encryption to aggregate quantities of supply and demand at the auctioneer's side. We then show how to tackle concerns that have the potential to disrupt the correctness of the auction procedure, such as negative quantities, leaving out offers of specific traders, or inserting offers after the auction has closed, using Zero-knowledge proofs and commitments. We implement our auction scheme in Rust and evaluate the performance of the individual steps. Our results show that the scheme is viable in runtime on consumer hardware for a reasonable number of offers. We believe this work shows the potential of a privacy-preserving and verifiable double auction in the real world for applications such as electricity trading.

References

[1] L. Fine, *Auctions*, 2018. [Online]. Available: https://www.econlib.org/library/Enc/Auctions.html (visited on 12/03/2021).

[2] F. Block, *Global Online-Only Auction Sales Surpassed US$1 Billion in 2020*, 2021. [Online]. Available: https://www.barrons.com/articles/global-online-only-auction-sales-surpassed-us-1-billion-in-2020-01610580341 (visited on 05/05/2022).

[3] A. Ganti, *Insider Trading*, 2022. [Online]. Available: https://www.investopedia.com/terms/i/insidertrading.asp (visited on 05/13/2022).

[4] S. Parsons, M. Marcinkiewicz, J. Niu, and S. Phelps, "Everything you wanted to know about double auctions , but were afraid to (bid or) ask," City University of New York, 2006.

[5] J. Fernando, *Law of Supply and Demand*, 2021. [Online]. Available: https://www.investopedia.com/terms/l/law-of-supply-demand.asp (visited on 05/05/2022).

[6] C. Halton, *Walrasian Market*, 2021. [Online]. Available: https://www.investopedia.com/terms/w/walrasian.asp (visited on 05/05/2022).

[7] D. Shah and S. Chatterjee, "A comprehensive review on day-ahead electricity market and important features of world's major electric power exchanges," *International Transactions on Electrical Energy Systems*, vol. 30, no. 7, e12360, 2020. DOI: 10.1002/2050-7038.12360.

[8] M. Babaioff and N. Nisan, "Concurrent Auctions Across The Supply Chain," *Journal of Artificial Intelligence Research*, vol. 21, pp. 595–629, May 2004, ISSN: 1076-9757. DOI: 10.1613/jair.1316.

[9] G. Edelman, *Google's Alleged Scheme to Corner the Online Ad Market*, 2021. [Online]. Available: https://www.wired.com/story/google-antitrust-ad-market-lawsuit/ (visited on 02/18/2022).

[10] OECD, *Preventing Corruption in Public Procurement*, 2016. [Online]. Available: https://www.oecd.org/gov/ethics/Corruption-Public-Procurement-Brochure.pdf (visited on 05/26/2022).

[11] European Commision, *Antitrust: Commission opens investigation into possible anticompetitive behaviour by the power exchange EPEX Spot*, 2021. [Online]. Available: https://ec.europa.eu/commission/presscorner/detail/en/ip_21_1523 (visited on 02/21/2022).

[12] P. Bogetoft *et al.*, "Secure Multiparty Computation Goes Live," *Lecture Notes in Computer Science*, pp. 325–343, 2009, ISSN: 0302-9743. DOI: 10.1007/978-3-642-03549-4_20.

[13] EPEX Spot, *Epex spot*, 2022. [Online]. Available: https://www.epexspot.com (visited on 02/21/2022).

[14] Nordpool Group, *Nord Pool*, 2022. [Online]. Available: https://www.nordpoolgroup.com (visited on 02/21/2022).

[15] European Union, "Regulation (EU) No. 1227/2011 of the European Parliament and of the Council of 25 October 2011 on wholesale energy market integrity and transparency (REMIT)," *Official Journal of the European Union*, vol. L326, pp. 1–16, 2011.

[16] J. Cartlidge, N. P. Smart, and Y. Talibi Alaoui, "MPC Joins The Dark Side," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, ser. Asia CCS '19, Auckland, New Zealand: Association for Computing Machinery, 2019, pp. 148–159, ISBN: 9781450367523. DOI: 10.1145/3321705.3329809.

[17] J. Cartlidge, N. P. Smart, and Y. Talibi Alaoui, "Multi-party computation mechanism for anonymous equity block trading: A secure implementation of turquoise plato uncross," *Intelligent Systems in Accounting, Finance and Management*, vol. 28, no. 4, pp. 239–267, 2021. DOI: 10.1002/isaf.1502.

[18] Partisia, *Secure Multiparty Computation Goes Live*, 2021. [Online]. Available: https://partisia.com/better-market-solutions/mpc-goes-live/ (visited on 02/22/2022).

[19] M. Franklin and M. Reiter, "The design and implementation of a secure auction service," *IEEE Transactions on Software Engineering*, vol. 22, no. 5, pp. 302–312, May 1996, ISSN: 1939-3520. DOI: 10.1109/32.502223.

[20] H. S. Galal and A. M. Youssef, "Succinctly Verifiable Sealed-Bid Auction Smart Contract," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, J. Garcia-Alfaro, J. Herrera-Joancomartí, G. Livraga, and R. Rios, Eds., Cham: Springer International Publishing, 2018, pp. 3–19, ISBN: 978-3-030-00305-0. DOI: 10.1007/978-3-030-00305-0_1.

[21] R. Sarenche, M. Salmasizadeh, M. H. Ameri, and M. R. Aref, "A secure and privacy-preserving protocol for holding double auctions in smart grid," *Information Sciences*, vol. 557, pp. 108–129, 2021, ISSN: 0020-0255. DOI: 10.1016/j.ins.2020.12.038.

[22] N. P. Smart, *Cryptography Made Simple*, 1st. Springer Publishing Company, Incorporated, 2015, ISBN: 3319219359.

[23] E. Barker, "Recommendation for Key Managements: Part 1 - General," National Institute of Standards and Technology, Tech. Rep. NIST Special Publication (SP) 800-57, Part 1, Rev. 5, 2020. DOI: 10.6028/NIST.SP.800-57pt1r5.

[24] E. Makri, *Co6gc: Linear secret sharing schemes - lsss*, 2020. [Online]. Available: https://www.esat.kuleuven.be/cosic/blog/lsss/ (visited on 05/20/2022).

[25] L. Meier, *Explaining yao's garbled circuits*, 2022. [Online]. Available: https://cronokirby.com/posts/2022/05/explaining-yaos-garbled-circuits/ (visited on 05/20/2022).

[26] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*, 2nd. Chapman & Hall/CRC, 2014, ISBN: 1466570261.

[27] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Advances in Cryptology — EUROCRYPT '99*, J. Stern, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238, ISBN: 978-3-540-48910-8. DOI: 10.1007/3-540-48910-X_16.

[28] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," in *Advances in Cryptology*, G. R. Blakley and D. Chaum, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 10–18, ISBN: 978-3-540-39568-3.

[29] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology — CRYPTO '91*, J. Feigenbaum, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 129–140, ISBN: 978-3-540-46766-3.

[30] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," in *Cryptographic Hardware and Embedded Systems – CHES 2011*, B. Preneel and T. Takagi, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 124–142, ISBN: 978-3-642-23951-9.

[31] C. P. Schnorr, "Efficient Identification and Signatures for Smart Cards," in *Advances in Cryptology — CRYPTO' 89 Proceedings*, G. Brassard, Ed., New York, NY: Springer New York, 1990, pp. 239–252, ISBN: 978-0-387-34805-6.

[32] Wikipedia contributors, *Zero-knowledge proof — Wikipedia, the free encyclopedia*, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Zero-knowledge_proof&oldid=1077304875 (visited on 03/28/2022).

[33] A. Fiat and A. Shamir, "How To Prove Yourself: Practical Solutions to Identification and Signature Problems," in *Advances in Cryptology — CRYPTO' 86*, A. M. Odlyzko, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194, ISBN: 978-3-540-47721-1. DOI: 10.1007/3-540-47721-7_12.

[34] G. Bleumer, "Random Oracle Model," in *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg and S. Jajodia, Eds. Boston, MA: Springer US, 2011, pp. 1027–1028, ISBN: 978-1-4419-5906-5. DOI: 10.1007/978-1-4419-5906-5_220.

[35] D. Bernhard, O. Pereira, and B. Warinschi, *How not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios*, Cryptology ePrint Archive, Report 2016/771, https://ia.cr/2016/771, 2016.

[36] B. Schoenmakers, *Lecture Notes Cryptographic Protocols*, 2022. [Online]. Available: https://www.win.tue.nl/~berry/CryptographicProtocols/LectureNotes.pdf.

[37] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short Proofs for Confidential Transactions and More," in *2018 IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020.

[38] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, "Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting," in *Advances in Cryptology – EUROCRYPT 2016*, M. Fischlin and J.-S. Coron, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 327–357, ISBN: 978-3-662-49896-5. DOI: 10.1007/978-3-662-49896-5_12.

[39] A. R. Choudhuri, M. Green, A. Jain, G. Kaptchuk, and I. Miers, *Fairness in an Unfair World: Fair Multiparty Computation from public Bulletin Boards*, Cryptology ePrint Archive, Paper 2017/1091, https://eprint.iacr.org/2017/1091, 2017. [Online]. Available: https://eprint.iacr.org/2017/1091.

[40] Google, *Certificate transparency*, 2022. [Online]. Available: https://certificate.transparency.dev/ (visited on 06/20/2022).

[41] A. C. Yao, "Protocols for secure computations," in *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, Nov. 1982, pp. 160–164. DOI: 10.1109/SFCS.1982.38.

[42] R. Alvarez and M. Nojoumian, "Comprehensive survey on privacy-preserving protocols for sealed-bid auctions," *Computers & Security*, vol. 88, p. 101 502, 2020, ISSN: 0167-4048. DOI: 10.1016/j.cose.2019.03.023.

[43] K. Suzuki, K. Kobayashi, and H. Morita, "Efficient Sealed-bid Auction using Hash Chain," in *Information Security and Cryptology — ICISC 2000*, D. Won, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 183–191, ISBN: 978-3-540-45247-8. DOI: 10.1007/3-540-45247-8_15.

[44] F. Brandt, "Cryptographic Protocols for Secure Second-Price Auctions," in *Cooperative Information Agents V*, M. Klusch and F. Zambonelli, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 154–165, ISBN: 978-3-540-44799-3. DOI: 10.1007/3-540-44799-7_16.

[45] M. Król, A. Sonnino, A. Tasiopoulos, I. Psaras, and E. Rivière, "Pastrami: Privacy-preserving, auditable, scalable & trustworthy auctions for multiple items," in *Proceedings of the 21st International Middleware Conference*, ser. Middleware '20, Delft, Netherlands: Association for Computing Machinery, 2020, pp. 296–310, ISBN: 9781450381536. DOI: 10.1145/3423211.3425669.

[46] A. Abidin, A. Aly, S. Cleemput, and M. A. Mustafa, "An MPC-Based Privacy-Preserving Protocol for a Local Electricity Trading Market," in *Cryptology and Network Security*, S. Foresti and G. Persiano, Eds., Cham: Springer International Publishing, 2016, pp. 615–625, ISBN: 978-3-319-48965-0. DOI: 10.1007/978-3-319-48965-0_40.

[47] J. R. Wallrabenstein and C. Clifton, "Privacy Preserving Tâtonnement," in *Financial Cryptography and Data Security*, N. Christin and R. Safavi-Naini, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 399–416, ISBN: 978-3-662-45472-5. DOI: 10.1007/978-3-662-45472-5_26.

[48] L. Liu, M. Du, and X. Ma, "Blockchain-Based Fair and Secure Electronic Double Auction Protocol," *IEEE Intelligent Systems*, vol. 35, no. 3, pp. 31–40, May 2020, ISSN: 1941-1294. DOI: 10.1109/MIS.2020.2977896.

[49] C. Wang, H.-f. Leung, and Y. Wang, "Secure Double Auction Protocols with Full Privacy Protection," in *Information Security and Cryptology - ICISC 2003*, J.-I. Lim and D.-H. Lee, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 215–229, ISBN: 978-3-540-24691-6. DOI: 10.1007/978-3-540-24691-6_17.

[50] B. Liu, S. Xie, Y. Yang, R. Wang, and Y. Hong, "Privacy preserving divisible double auction with a hybridized TEE-blockchain system," *Cybersecurity*, vol. 4, no. 1, p. 37, 2021. DOI: 10.1186/s42400-021-00100-x.

[51] Y. Xu, Z. Chen, and H. Zhong, "Privacy-preserving Double Auction Mechanism Based on Homomorphic Encryption and Sorting Networks," *ArXiv*, vol. abs/1909.07637, 2019. DOI: 10.48550/ARXIV.1909.07637.

[52] H. S. Galal and A. M. Youssef, "Publicly Verifiable and Secrecy Preserving Periodic Auctions," in *Financial Cryptography and Data Security. FC 2021 International Workshops*, M. Bernhard *et al.*, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2021, pp. 348–363, ISBN: 978-3-662-63958-0. DOI: 10.1007/978-3-662-63958-0_29.

[53] R. P. McAfee, "A dominant strategy double auction," *Journal of Economic Theory*, vol. 56, no. 2, pp. 434–450, 1992, ISSN: 0022-0531. DOI: 10.1016/0022-0531(92)90091-U.

[54] A. Nilsson, P. N. Bideh, and J. Brorsson, "A Survey of Published Attacks on Intel SGX," *ArXiv*, vol. abs/2006.13598, 2020. DOI: 10.48550/ARXIV.2006.13598.

[55] B. Toulas, *New Intel chips won't play Blu-ray disks due to SGX deprecation*, 2022. [Online]. Available: https://www.bleepingcomputer.com/news/security/new-intel-chips-wont-play-blu-ray-disks-due-to-sgx-deprecation/ (visited on 03/04/2022).

[56] Z. Chen, R. Che, H. Zhong, M. Tian, and J. Cui, "PATH: privacy-preserving auction for heterogeneous spectrum allocations," *Wireless Networks*, vol. 25, no. 4, pp. 1763–1776, 2019, ISSN: 1022-0038. DOI: 10.1007/s11276-017-1628-5.

[57] Z. Chen, X. Wei, H. Zhong, J. Cui, Y. Xu, and S. Zhang, “Secure, efficient and practical double spectrum auction,” in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, Jun. 2017, pp. 1–6. DOI: 10.1109/IWQoS.2017.7969153.

[58] F. Hu, B. Chen, J. Wang, M. Li, P. Li, and M. Pan, “MastDP: Matching Based Double Auction Mechanism for Spectrum Trading with Differential Privacy,” in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2019, pp. 1–6. DOI: 10.1109/GLOBECOM38437.2019.9013917.

[59] Q. Wang, J. Huang, Y. Chen, X. Tian, and Q. Zhang, “Privacy-Preserving and Truthful Double Auction for Heterogeneous Spectrum,” *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 848–861, Apr. 2019, ISSN: 1558-2566. DOI: 10.1109/TNET.2019.2903879.

[60] S. Xie, H. Wang, Y. Hong, and M. Thai, “Privacy Preserving Distributed Energy Trading,” in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, Los Alamitos, CA, USA: IEEE Computer Society, Dec. 2020, pp. 322–332. DOI: 10.1109/ICDCS47774.2020.00078.

[61] T. Gaybullaev, H.-Y. Kwon, T. Kim, and M.-K. Lee, “Efficient and Privacy-Preserving Energy Trading on Blockchain Using Dual Binary Encoding for Inner Product Encryption,” *Sensors*, vol. 21, no. 6, 2021, ISSN: 1424-8220. DOI: 10.3390/s21062024.

[62] N. Z. Aitzhan and D. Svetinovic, “Security and Privacy in Decentralized Energy Trading Through Multi-Signatures, Blockchain and Anonymous Messaging Streams,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 840–852, Sep. 2018, ISSN: 1941-0018. DOI: 10.1109/TDSC.2016.2616861.

[63] A. Samy, H. Yu, H. Zhang, and G. Zhang, “SPETS: Secure and Privacy-Preserving Energy Trading System in Microgrid,” *Sensors*, vol. 21, no. 23, 2021, ISSN: 1424-8220. DOI: 10.3390/s21238121.

[64] K. Peng, C. Boyd, E. Dawson, and K. Viswanathan, “Robust, Privacy Protecting and Publicly Verifiable Sealed-Bid Auction,” in *Information and Communications Security*, R. Deng, F. Bao, J. Zhou, and S. Qing, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 147–159, ISBN: 978-3-540-36159-6. DOI: 10.1007/3-540-36159-6_13.

[65] C. Wang and H.-f. Leung, “Anonymity and security in continuous double auctions for Internet retails market,” in *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, Jan. 2004, 10 pp.-. DOI: 10.1109/HICSS.2004.1265431.

[66] M. Nateghizad, Z. Erkin, and R. L. Lagendijk, “An efficient privacy-preserving comparison protocol in smart metering systems,” *EURASIP Journal on Information Security*, vol. 2016, no. 1, p. 11, 2016. DOI: 10.1186/s13635-016-0033-4.

[67] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, “Privacy-preserving face recognition,” in *Privacy Enhancing Technologies*, I. Goldberg and M. J. Atallah, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 235–253, ISBN: 978-3-642-03168-7.

[68] I. Damgård, M. Geisler, and M. Krøigaard, “Efficient and secure comparison for on-line auctions,” in *Information Security and Privacy*, J. Pieprzyk, H. Ghodosi, and E. Dawson, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 416–430, ISBN: 978-3-540-73458-1.

[69] M. J. Jurik, “Extensions to the Paillier Cryptosystem with Applications to Cryptological Protocols,” Ph.D. dissertation, University of Aarhus, 2003.

[70] I. Lovecruft, *Ed25519-dalek*, 2020. [Online]. Available: https://crates.io/crates/ed25519-dalek (visited on 06/06/2022).

[71] H. d. Valence, *Curve25519-dalek-ng*, 2021. [Online]. Available: https://crates.io/crates/curve25519-dalek-ng (visited on 06/06/2022).

[72] J. Vos, *Scicrypt*, 2022. [Online]. Available: https://crates.io/crates/scicrypt (visited on 06/02/2022).

[73] H. d. Valence, *Bulletproofs*, 2021. [Online]. Available: https://crates.io/crates/bulletproofs (visited on 06/06/2022).

[74] J.-H. Hoepman, *Privacy is hard*, 2021. [Online]. Available: https://blog.xot.nl/category/privacy-is-hard/index.html (visited on 06/13/2022).

[75] C. Cadwalladr and E. Graham-Harrison, *Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach*, 2018. [Online]. Available: https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election (visited on 06/14/2022).

[76] T. Nishide and K. Sakurai, “Distributed paillier cryptosystem without trusted dealer,” in *Information Security Applications*, Y. Chung and M. Yung, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 44–60, ISBN: 978-3-642-17955-6.

[77] K. Cheng, W. Tong, L. Zheng, J. Fu, X. Mu, and Y. Shen, “A Secure and Fair Double Auction Framework for Cloud Virtual Machines,” *IEEE Access*, vol. 9, pp. 87 982–87 994, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3089492.