Privacy-Preserving Multi-Party Bartering Secure Against Active Adversaries

Stefan Wüller[1], Ulrike Meyer[1], and Susanne Wetzel[2]

[1] RWTH Aachen University, Aachen, Germany
{wueller,meyer}@itsec.rwth-aachen.de
[2] Stevens Institute of Technology, Hoboken, NJ, USA
swetzel@stevens.edu

Abstract. A majority of electronic bartering transactions is carried out via online platforms. Typically, these platforms require users to disclose sensitive information about their trade capabilities which might restrict their room for negotiation. It is in this context that we propose a novel decentralized and privacy-preserving bartering protocol for multiple parties that offers the same privacy guarantees as provided by traditional bartering and by cash payments. The proposed protocol is even secure against an active attacker who controls a majority of colluding parties.

1 Introduction

Bartering—the cashless act of exchanging goods and services—has been practiced since the early days of humanity. The traditional form of bartering requires a party to find another party such that their offers and demands align. Consider the following example from [13]: Alice demands a tool while Bob demands some medicine. For the case that each of them offers what the other party demands, both parties are lucky since they can barter. But for the case that Alice offers food instead of medicine, they have to find a larger trade cycle where more than two parties exchange their offered commodities in a cyclic fashion. So, Alice and Bob could try to find a third party, Carol, who is willing to exchange medicine for food. As this simple example already shows, the major drawback of traditional bartering is the inevitable need for coordination, i.e., a group of parties whose offers and demands align have to be in the same place at the same time [13]. In modern economies, this coordination issue was resolved by the introduction of currencies which serve as a mediator and thus eliminate the need for the explicit search of trade cycles.

Nevertheless, in recent years bartering became popular again [11]. This is also due to the emerging of online bartering platforms such as *U-Exchange*, *BarterQuest*, or *TradeYa* [18, 1, 17] which typically provide a variety of functionalities facilitating the bartering process (including the cumbersome search for trade partners) between individuals as well as businesses. These functionalities comprise supporting the specification of offers and demands; searching, browsing, and filtering the offers of other users; searching for trade partners in some

automated fashion; or supporting negotiating of the final exchange rates. However, an inherent requirement of these platforms is that a user has to disclose her offered and demanded commodities (along with the corresponding quantities) to at least the operator and oftentimes to all other users—even those who do not qualify as trade partners.

The goal of this paper is to devise a bartering process that offers the same privacy guarantees ideally provided by traditional bartering or traditional cash currencies. More precisely, we require that a party only learns what it receives and what it has to send (and what can be deduced from it). In particular, a party does not learn anything about the capabilities and the activities of parties it does not directly trade with. Our solution does not require a (trusted) third party to observe or coordinate the bartering transactions.

One of the first steps in this direction was proposed in [21, 22]. The authors devise a privacy-preserving multi-party bartering protocol which is based on *secure multi-party computation* (SMPC) techniques. However, this protocol only provides security in the semi-honest model, i.e., under the strict assumption that all parties follow the protocol specification but try to learn as much information as possible from participating in the protocol. We extend on the protocol in [21, 22] and propose a privacy-preserving multi-party bartering protocol which provides security against malicious (i.e., active) adversaries. Analogously to [21, 22], our novel bartering protocol allows a party to specify a quote indicating its offered and desired commodity as well as the corresponding quantity ranges at which the party is willing to trade. Given a set of parties along with their private quotes, our protocol automatically and securely determines the actual trade comprising the trade partner constellation (i.e., who trades with whom) as well as the commodities and quantities to be traded. A party's quote remains private throughout the entire protocol execution. From participating in the bartering protocol, a party merely learns its local view consisting of its direct trade partners and the quantities of the commodities to be sent and received. Our protocol is able to incorporate any selection strategy for selecting the trade partner constellation as long as the strategy is a function of the considered trade partner constellations. Examples for a selection strategy one may choose include the maximization of the number of parties able to trade, the minimization of the length of the trade cycles in the actual trade, as well as a combination of both. Furthermore, our protocol includes a privacy-preserving mechanism to replace a manual negotiation of the final exchange rates. This mechanism is implemented by sampling the actual quantities out of private intervals where the boundaries of such an interval are determined from the negotiation ranges specified by the involved parties. Our privacy-preserving bartering protocol allows the sampling of the quantities according to an arbitrary discrete distribution. Note that for the case that the negotiation of the quantities is left to the parties, a party which, e.g., has to make the first offer or which has the least bargaining experience may be discriminated. Our approach motivates the parties to honestly specify their negotiation ranges.

Contributions: Building on [21, 22], we propose the first privacy-preserving multi-party bartering protocol which provides security against malicious adversaries. Due to the special purpose design of the bartering protocol presented in [21, 22], a straight-forward transformation to the malicious model is not possible. Instead, new design approaches, e.g., for restricting a party's local view and for the integration of a negotiation mechanism, are required. At the core of our bartering protocol are two novel building blocks which also provide security in the malicious model. First, we introduce a privacy-preserving building block implementing the *conditional random selection* (CRS) functionality which has been introduced in [20]. CRS allows to obliviously select one random data entry which satisfies a specified condition out of a private set of data entries and can be used in the context of bartering for determining an actual trade [21, 22]. Second, we introduce a building block for randomly sampling elements out of a private interval (given by encrypted interval boundaries only) w.r.t. an arbitrary discrete distribution. In the context of bartering, we use this protocol to have a flexible means for determining trade rates. These novel building blocks are of general interest beyond the context of bartering. While the complexity of the protocol from [21, 22] linearly depends on the cardinality of the commodity space, our novel design approach results in a protocol with complexity independent of the number of supported commodities.[3]

Outline: The remainder of this paper is organized as follows. After reviewing related work in Section 2, we provide an overview on designing SMPC protocols for the malicious model (Section 3). Subsequently, we devise an intuitive terminology for multi-party bartering based on graphs and provide an intuition for the functionalities of our novel protocol (Section 4). In Section 5, we review existing building blocks used in the context of our work and present two novel building blocks for conditional random selection and for random sampling from a private interval according to an arbitrary discrete distribution. Building on that, we introduce our novel privacy-preserving bartering protocol (Section 6). The paper closes with some remarks on future work.

2 Related Work

Our work models bartering in the same way as the privacy-preserving multi-party bartering protocol which is secure against semi-honest adversaries introduced by Wüller et al. [21, 22]. For a given set of parties and their quotes, the bartering protocol allows to securely determine an actual trade comprising the trade constellation of the parties as well as the commodities and quantities to be traded. Our work presented in this paper extends their setting in that we provide security against malicious adversaries. This requires the design of a new bartering protocol composed of new building blocks which are of general interest. While in [21, 22] only a basic level of fairness w.r.t. the selection of the quantities to be traded

[3] Note that the novel design approach can also be applied to [21, 22] in order to improve the efficiency.

is considered, in this paper, we extend the notion of fairness by providing an opportunity to reduce the probability of imbalanced determined quantities (for details see Section 6). Note that the extended notation of fairness can directly be applied to the two-party bartering protocol presented in [23].

To the best of our knowledge, to date there are no other privacy-preserving bartering approaches for more than two parties which provide security against malicious adversaries. For the two-party case, a privacy-preserving bartering protocol has been introduced in [23]. However, privacy-preserving bartering protocols for the two-party case cannot be used for the much more complicated bartering setting with more than two parties. This is due to the fact that these protocols are not capable of determining trade cycles between more than two parties [21].

As detailed in [12], bartering transactions offer a richer structure of exchanges compared to transaction in the context of e-commerce (and auctions). Consequently, privacy-preserving protocols for e-commerce scenarios or auctions can not be directly applied for implementing a privacy-preserving bartering protocol.

3 Preliminaries

In the following, we first recap general notation, the Paillier cryptosystem, as well as the definition of security we use throughout the paper. Then, we provide a brief overview of the SMPC framework from [3, 4] which we utilize to design protocols secure against malicious adversaries.

3.1 Notation and Definitions

To indicate that a is drawn uniformly at random from A, we write $a \leftarrow_{\$} A$. $\mathbb{N}_u$ refers to the set of natural numbers less than or equal to $u \in \mathbb{N}$. $L[i]$ refers to entry l_i of vector $L = (l_1, \ldots, l_n)$ with $i \in \mathbb{N}_n$. For some predicate B, let $[B]$ denote the *Iverson bracket* where $[B] = 1$ if B is true and otherwise $[B] = 0$. Given an integer interval $[\lambda, \mu]$ with $\lambda, \mu \in \mathbb{N}$, the width of an interval, written $|[\lambda, \mu]|$ is defined as $\mu - \lambda$. We denote the index set of all parties $P_1, \ldots, P_\iota$ ($\iota \in \mathbb{N}$) participating in a multi-party protocol as $\mathscr{P} := \{1, \ldots, \iota\}$.

3.2 Paillier Threshold Cryptosystem

The Paillier cryptosystem [14] is an additively homomorphic encryption scheme which provides for semantic security. In order to design privacy-preserving protocols, we use the (τ, ι) threshold variant of Paillier introduced in [7] where the private decryption key is distributed amongst ι parties such that τ parties ($\tau \leq \iota$) have to cooperate in order to decrypt a ciphertext. In the following, we outline the key generation and the encryption function of this threshold variant of the Paillier cryptosystem.

Key Generation: For a given security parameter s, choose two primes p and q of bit length $s/2$ such that there exist two primes p' and q' with $p = 2p' + 1$ and $q = 2q' + 1$ and compute $N := p \cdot q$. Set $N' := p'q'$, $\beta \leftarrow_\$ \mathbb{Z}_N^*$, $(a, b) \leftarrow_\$ \mathbb{Z}_N^* \times \mathbb{Z}_N^*$, and $g := (1+N)^a \cdot b^N \mod N^2$. The private key $\beta N'$ is shared among ι parties using Shamir's (τ, ι) secret sharing scheme [16]. The public key is the same for all ι parties and consists of g, N, and $\Theta := L(g^{N'\beta}) = aN'\beta \mod N$, where $L(x) := (x-1)/N$.

Encryption: A message m in *plaintext space* $\mathbb{P} := \mathbb{Z}_N$ is computed by selecting $r \leftarrow_\$ \mathbb{Z}_N^*$ and computing $c = E(m) := g^m r^N \mod N^2$ where the resulting ciphertext c is an element of the *ciphertext space* $\mathbb{C} := \mathbb{Z}_{N^2}^*$.

For matters of convenience, we omit the public and private key from our notation. $\mathbb{P}$ forms an additive group $(\mathbb{Z}_N, +)$ and $\mathbb{C}$ forms a multiplicative group $(\mathbb{Z}_{N^2}^*, \cdot)$. Let $m, m_1, m_2 \in \mathbb{P}$ and $a \in \mathbb{Z} \backslash \{0\}$. Then, the $((\tau, \iota)$ threshold) Paillier cryptosystem provides for *homomorphic addition*

$$E(m_1) +_h E(m_2) := E(m_1) \cdot E(m_2)$$

such that $D(E(m_1) \cdot E(m_2)) = D(E(m_1 + m_2))$ and *homomorphic scalar multiplication*

$$E(m) \times_h a := (E(m))^a$$

such that $D((E(m))^a) = D(E(a \cdot m))$. We define $E(m) \times_h 0 := E(0)$ and $E(m_1) -_h E(m_2) := E(m_1) +_h (E(m_2))^{-1}$ such that $D(E(m_1) +_h (E(m_2))^{-1}) = D(E(m_1 - m_2))$. A ciphertext c can be randomized by computing $Rnd(c) := c +_h E(0)$ where $E(0)$ is a fresh encryption of 0. Given two ciphertexts $E(m_1)$, $E(m_2)$ and knowing the upper bound for the number $d \in \mathbb{N}$ of decimal places of m_2 with $(m_1 \cdot 10^d + m_2) < N$, we define the *homomorphic concatenation* of two ciphertexts as

$$E(m_1) \mid\mid_h E(m_2) := E(m_1) \times_h 10^d +_h E(m_2).$$

In the following, we write $[\![m]\!] := E(m)$ in order to refer to an encryption of a plaintext m.

3.3 Secure Multi-Party Computation

Secure multi-party computation allows a set of ι parties to compute an ι-input functionality $\mathcal{F}$ such that each participating party learns nothing beyond its prescribed output and what can be deduced from it in combination with its private input. The security (i.e., privacy and correctness) of the computation is guaranteed even in the presence of an adversary controlling a fixed set of parties.

Let $\overline{x} := (x_1, \ldots, x_\iota)$ and let $\mathcal{F} : (\{0,1\}^*)^\iota \to (\{0,1\}^*)^\iota, \overline{x} \mapsto (\mathcal{F}_1(\overline{x}), \ldots, \mathcal{F}_\iota(\overline{x}))$ be an ι-input functionality where P_v ($v \in \mathscr{P}$) provides its private input $x_v \in \{0,1\}^*$ and obtains its prescribed (private) output $\mathcal{F}_v(\overline{x})$. Let π be an ι-party protocol implementing functionality $\mathcal{F}$. We write $I_C = \{i_1, ..., i_t\} \subset \mathscr{P}$ for

the index set of $t < \iota$ corrupted parties controlled by the adversary. We consider a malicious adversary which may instruct the corrupted parties to arbitrarily deviate from the protocol specification.

In the SMPC model from [2], security is stated in terms of the ideal world vs. the real world paradigm. The capabilities of an adversary $\mathfrak{A}$ in the *real world* execution of π are compared to the capabilities of an adversary $\mathfrak{S}$ interacting in the *ideal world* where the parties have access to a trusted third party which computes the target functionality $\mathcal{F}$ and provides each party with its prescribed output. Consequently, the computation of $\mathcal{F}$ in the ideal world achieves the highest level of security. In the real world, the parties do not have access to a trusted third party but have to execute protocol π in order to obtain their respective outputs. In the following, the ideal world (resp., real world) *view* of a party refers to the information (e.g., the received messages) it learns during the computation of $\mathcal{F}$ (resp., execution of π).

The random variable $\text{IDEAL}_{\mathcal{F},\mathfrak{S}}(\overline{x}, I_C, k, a)$ denotes the $\iota + 1$ tuple which includes the output of all ι parties and the output of the adversary $\mathfrak{S}$ for input $\overline{x}$, the index set of corrupted parties I_C, security parameter k, and auxiliary input $a \in \{0,1\}^*$. Functionality $\mathcal{F}$ is computed in the ideal world under the attack of adversary $\mathfrak{S}$. Analogously, the random variable $\text{REAL}_{\pi,\mathfrak{A}}(\overline{x}, I_C, k, a)$ denotes the $\iota + 1$ tuple which is comprised of the outputs where π is executed in the real world under the attack of adversary $\mathfrak{A}$.

Definition 1. (Security in the Malicious Model.) *Let π be a ι-party protocol implementing the ι-party functionality $\mathcal{F}$. Protocol π securely implements $\mathcal{F}$ in the malicious model iff for any static real world adversary $\mathfrak{A}$ which corrupts a subset I_C of t parties there exists an ideal world adversary $\mathfrak{S}$ such that*

$$\begin{aligned} IDEAL_{\mathcal{F},\mathfrak{S}} := \; & \{IDEAL_{\mathcal{F},\mathfrak{S}}(\overline{x}, I_C, k, a)\}_{\overline{x}, I_C, k, a} \stackrel{c}{\equiv} \\ & \{REAL_{\pi,\mathfrak{A}}(\overline{x}, I_C, k, a)\}_{\overline{x}, I_C, k, a} =: REAL_{\pi,\mathfrak{A}} \end{aligned}$$

where $\stackrel{c}{\equiv}$ denotes computational indistinguishability.

3.4 CDN Framework for Secure Computations

Our novel privacy-preserving bartering protocol is designed on top of the CDN framework [4, 3] which allows ι parties to securely compute a protocol π (implementing functionality $\mathcal{F}$) in the presence of a malicious adversary who controls at most a minority of $\tau < \iota/2$ corrupted parties. Security is defined according to Definition 1 with the restriction that $|I_C| < \lceil \iota/2 \rceil$. Furthermore, each party is allowed to additionally receive a public input and output. Assuming an honest majority guarantees the termination of a protocol execution with the correct computation result. The CDN-framework assumes a threshold homomorphic cryptosystem satisfying a set of specific properties. The threshold variant of the Paillier cryptosystem presented in Section 3.2 satisfies these properties and will be used in the remainder of the paper. In order to prevent a corrupted party from cheating, each party has to prove (by the means of zero knowledge

proofs) that it follows the protocol. Once a party is found to deviate from the protocol specification it is excluded from the further execution of the protocol. For the sake of brevity, we will not explicitly address this case in the following sections.

At the core of the CDN framework there is a universal protocol $\text{FuncEval}_{\mathcal{F}}$ which expects the description of an arithmetic circuit implementing functionality $\mathcal{F}$. The circuit has to be composed of some basic gates (see below). A gate ρ is a kind of black box protocol characterized by the fact that it obtains encrypted input, specifies how to compute some gate functionality $\mathcal{G}$ on that input, and returns the encrypted result of the computation.

Loosely speaking, the evaluation of protocol $\text{FuncEval}_{\mathcal{F}}$ can be divided into the following three phases [23]:

Input Phase: All parties encrypt their private inputs and broadcast the corresponding ciphertext(s). Additionally, for each broadcasted ciphertext a party proves in zero knowledge that it knows the corresponding plaintext.

Computation Phase: All parties jointly evaluate the given circuit gate by gate. Each gate obtains an encrypted input and provides an encrypted output which in turn may constitute the input for another gate.

Output Phase: All parties who are still participating in the protocol jointly execute a private decryption protocol providing each party with its prescribed output.

We write $([\![o]\!]) \leftarrow \mathcal{G}([\![x]\!])$ (resp., $([\![o]\!]) \leftarrow \rho([\![x]\!])$) to indicate that all parties have common encrypted input $[\![x]\!]$ and common encrypted output $[\![o]\!]$.

One of the basic gates proposed in [3, 4] is a multiplication gate $([\![a \cdot b]\!]) \leftarrow \rho_{\text{Mult}}(([\![a]\!], [\![b]\!]))$ allowing to multiply two common ciphertexts $[\![a]\!]$ and $[\![b]\!]$ of plaintexts $a, b \in \mathbb{P}$ such that each party obtains the result $[\![a \cdot b]\!]$. The broadcast complexity of ρ_{Mult} is in $O(\iota s)$ for security parameter s while the round complexity is in $O(1)$.

Theorem 1. (cf. [4].) *The $\text{FuncEval}_{\mathcal{F}}$ protocol securely evaluates $\mathcal{F}$ in the presence of a malicious adversary controlling a minority of parties.*

Generalizations: In the literature, several generalizations have been proposed for the CDN framework: Although the original version of the CDN framework requires that $\mathcal{F}$ is a deterministic functionality and assumes an honest majority, Theorem 1 is still valid for computing probabilistic functionalities and allowing all but one parties being controlled by an adversary [3, 15, 5]. In the former case, it has to be assured that the random bits influencing the protocol output are jointly computed and remain encrypted throughout the protocol execution. In case an adversary controls a majority of parties, protocol termination can not be guaranteed.

Furthermore, there exists a sophisticated technique to integrate new gates into the CDN framework. Strictly speaking, the integration of a new gate requires

one to prove that the simulated view of all parties is statistically indistinguishable from the real world view of all parties (given the encrypted input and output). However, Schoenmakers et al. argue that it suffices to show that a gate can be simulated in a statistically indistinguishable manner for inputs of a special form [15, 8, 10]. This technique exploits the modularity of the proof of Theorem 1 in [3] which is centered around an intermediary distribution YAD^b which is a function of an encrypted bit b. More precisely, the proof shows that

$$\text{IDEAL}_{\mathcal{F},\mathfrak{S}} \stackrel{p}{\equiv} \text{YAD}^0 \stackrel{c}{\equiv} \text{YAD}^1 \stackrel{s}{\equiv} \text{REAL}_{\pi,\mathfrak{A}},$$

where $\stackrel{p}{\equiv}$ and $\stackrel{s}{\equiv}$ refer to perfect and statistical indistinguishability, respectively. A distinguisher for distributions $\text{IDEAL}_{\mathcal{F},\mathfrak{S}}$ and $\text{REAL}_{\pi,\mathfrak{A}}$ yields a distinguisher for distributions YAD^0 and YAD^1. Since the gap between YAD^0 and YAD^1 depends on the single encrypted bit b, the security of a protocol is reduced to the security of the underlying cryptosystem. Consequently, a gate obtaining input $[\![x]\!]$ can be simulated for input $[\![\tilde{x}]\!] = [\![(1-b)x^{\langle 0\rangle} + bx^{\langle 1\rangle}]\!]$ where $x^{\langle 0\rangle}$ and $x^{\langle 1\rangle}$ represent x in the YAD^0 and YAD^1 distributions, respectively. Note that the simulator is provided with $x^{\langle 0\rangle}$, $x^{\langle 1\rangle}$, and $[\![b]\!]$. More details on this proof technique are given in [15, 8, 10].

4 Overview

4.1 Bartering Terminology

For a set of parties, a trade generically indicates which party receives (or sends) which quantity of which commodity from (or to) which other party. We allow each party to specify one offered and one desired commodity and focus on so-called $(1:1)$ *trades* where each party either receives and sends nothing or receives some quantity of its desired commodity from at exactly one party and sends some quantity of its offered commodity to one party.

More specifically, we consider a set of ι parties $\{P_v : v \in \mathscr{P}\}$ with $\mathscr{P} := \mathbb{N}_\iota$ and a publicly known finite set $\mathscr{C} := \{c_1, \ldots, c_{|\mathscr{C}|}\}$ of divisible commodities. Each party P_v specifies exactly one *quote* $\mathbf{q}^{(v)} := (\mathbf{o}^{(v)}, \mathbf{d}^{(v)})$ where $\mathbf{o}^{(v)}$ and $\mathbf{d}^{(v)}$ are P_v's *offer* and *demand*, respectively. We model $\mathbf{o}^{(v)}$ as a 2-tuple $\mathbf{o}^{(v)} := (c_o^{(v)}, \overline{q}_o^{(v)})$ where $c_o^{(v)} \in \mathscr{C}$ specifies the commodity offered by P_v and $\overline{q}_o^{(v)} \in \mathbb{N}\backslash\{0\}$ denotes the maximum quantity at which $c_o^{(v)}$ is offered. Similarly, we model $\mathbf{d}^{(v)} := (c_d^{(v)}, \underline{q}_d^{(v)})$ with $c_d^{(v)} \in \mathscr{C}$ referring to P_v's desired commodity and $\underline{q}_d^{(v)} \in \mathbb{N}\backslash\{0\}$ indicating the minimum quantity at which this commodity is desired by P_v. With $\mathbf{q}^{(v)}$ a party P_v indicates that it is *satisfied* with a trade if it receives at least $\underline{q}_d^{(v)}$ units of commodity $c_d^{(v)}$ and sends at most $\overline{q}_o^{(v)}$ units of $c_o^{(v)}$. The *quantity ranges* of the offered and desired commodities of a party P_v are defined as $Q_o^{(v)} := [1, \overline{q}_o^{(v)}]$ and $Q_d^{(v)} := [\underline{q}_d^{(v)}, \infty]$. We write $q_{c_o^{(v)}}^{(v,v')}$ in order to indicate at which quantity $P_{v'}$ will receive commodity $c_o^{(v)}$ from P_v $(v, v' \in \mathscr{P},\ v \neq v')$.

G^{C}	Compatibility Graph	Def. 2
G^{TPC}	Trade Partner Constellation Graph	Def. 3
G^{PTPC}	Potential Trade Partner Constellation Graph	Def. 4
G^{AT}	Actual Trade Graph	Def. 6

Table 1: Summary of the main bartering terms.

To allow for an intuitive visualization, all further bartering terminology is defined in terms of graph theory. Figure 1 provides an example and shows how the terms build on each other.

For the set of quotes of all parties $\mathbf{Q} := \{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\iota)}\}$ a directed graph, referred to as *compatibility graph* G^{C}, can be constructed where each node is identified with a party index in $\mathscr{P}$. A directed edge (v, v') between two nodes v and v' indicates that condition $(c_o^{(v)} = c_d^{(v')}) \wedge (\overline{q}_o^{(v)} \geq \underline{q}_d^{(v')})$ is satisfied, i.e., that P_v can satisfy the demand of $P_{v'}$. Formally, a compatibility graph is defined as follows:

Definition 2. (Compatibility Graph.) *For a given set of quotes* $\mathbf{Q} = \{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\iota)}\}$ *with* $\mathbf{q}^{(v)} = ((c_o^{(v)}, \overline{q}_o^{(v)}), (c_d^{(v)}, \underline{q}_d^{(v)}))$ *with* $v \in \mathscr{P}$*, a* compatibility graph G^C *is a simple directed graph* (V, E) *with* $V := \{1, \ldots, \iota\}$ *and* $(v, v') \in E$ *iff* $[(c_o^{(v)} = c_d^{(v')}) \wedge (\overline{q}_o^{(v)} \geq \underline{q}_d^{(v')})] = 1$ *where* $v, v' \in \mathscr{P}, v \neq v'$.

Let $deg^+(v)$ $(v \in V)$ refer to the indegree of node v (resp., let $deg^-(v)$ refer to the outdegree of node v) and let $deg(v) := deg^+(v) + deg^-(v)$.

Similarly as for a compatibility graph, the nodes of a *trade partner constellation graph* G^{TPC} are identified with the (same) party indices in $\mathscr{P}$. However, the edges of a trade partner constellation graph are generic as well as independent of any set of quotes $\mathbf{Q}$ and for each node v it holds that it either has exactly one incoming and one outgoing edge (i.e., $deg^+(v) = deg^-(v) = 1$) or otherwise it is isolated (i.e., $deg(v) = 0$). Put differently, a trade partner constellation graph either indicates that a party P_v does not actively participate in the encoded constellation or specifies P_v's trade partners. In the latter case, G^{TPC} determines exactly one party from which P_v receives some quantity of a commodity (P_v's *offerer*) as well as exactly one party to which P_v has to send some quantity of a commodity (P_v's *demander*). The definition of a trade partner constellation graph ensures that each party which sends some quantity of a commodity in turn receives some quantity of another commodity.

Definition 3. (Trade Partner Constellation Graph.) *A* trade partner constellation graph G^{TPC} *is a simple directed graph* (V, E) *with* $V := \{1, \ldots, \iota\}$ *and* $\forall v \in V: \ (deg^-(v) = 1 \wedge deg^+(v) = 1) \vee (deg(v) = 0)$.

An m-cycle in a compatibility graph or in a trade partner constellation graph is referred to as an *m-trade cycle*. To ensure that commodities can be traded according to the direction of the edges in a trade partner constellation graph, it has

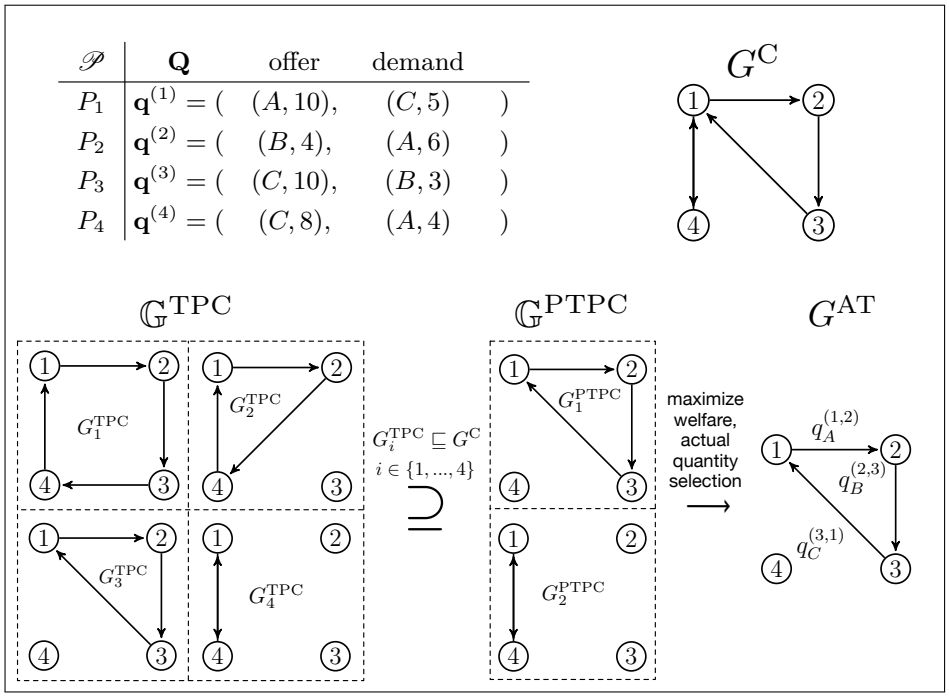

Fig. 1: Example for bartering terms and their relations.

to be assured that the trade cycles are disjoint (also referred to as *simultaneously executable*). While a compatibility graph may contain multiple trade cycles of which some may not be simultaneously executable (e.g., G^{C} in Figure 1 contains one 2-trade cycle and one 3-trade cycle but only one of them is executable at a time because P_1 is involved in both cycles), a potential trade constellation graph corresponds to a subgraph of either one trade cycle or multiple simultaneously executable trade cycles (e.g., G_1^{PTPC} and G_2^{PTPC} in Figure 1).

Definition 4. (Potential Trade Partner Constellation Graph.) *For a given set of quotes* $\mathbf{Q}$*, a trade partner constellation graph* G^{TPC} *is referred to as a* potential trade partner constellation graph G^{PTPC} *iff* G^{TPC} *is a subgraph of* G^{C}*, written* $G^{TPC} \sqsubseteq G^{C}$*.*

We write $\mathbb{G}^{\mathrm{TPC}} := \{G_1^{\mathrm{TPC}}, \ldots, G_{|\mathbb{G}^{\mathrm{TPC}}|}^{\mathrm{TPC}}\}$ for a set of trade partner constellation graphs. For a given $\mathbb{G}^{\mathrm{TPC}}$ and G^{C}, the set of potential trade partner constellation graphs is denoted as $\mathbb{G}^{\mathrm{PTPC}}$. According to Definition 4, it holds that $\mathbb{G}^{\mathrm{PTPC}} \subseteq \mathbb{G}^{\mathrm{TPC}}$ (cf. Figure 1).

Definition 5. (Welfare Function.) *A* welfare function $\mathcal{W}(\cdot) : \mathbb{G}^{TPC} \to \mathbb{N} \cup \{0\}$ *maps a trade partner constellation graph in* $\mathbb{G}^{TPC}$ *to a welfare.*

Definition 6. (Actual Trade Graph.) *For a given set of quotes* $\mathbf{Q}$ *and a welfare function* $\mathcal{W}$*, an* actual trade graph G^{AT} *corresponds to a potential trade partner constellation graph* G^{PTPC} *with maximum welfare drawn from* $\mathbb{G}^{PTPC}$*. Additionally, each edge* (v, v') *is associated with a weight* $q^{(v,v')}_{c_o^{(v)}}$ *which specifies the actual quantity at which the actual commodity* $c_o^{(v)}$ *is to be send from* P_v *to* $P_{v'}$*. We write* $G^{AT} \leftarrow_{max_{\mathcal{W}}} \mathbb{G}^{PTPC}$ *for the process of determining* G^{AT} *given* $\mathbb{G}^{PTPC}$*,* $\mathcal{W}$*, and* $\mathbf{Q}$*.*

For example, the welfare of a trade partner constellation graph $G^{\text{TPC}} = (V, E)$ can be defined as the number of its edges $\mathcal{W}(G^{\text{TPC}}) = |E|$ such that the selected actual trade graph maximizes the number of parties that can engage in the trade, i.e., exchange their commodities.

For matters of convenience, we define

$$N_o^{(v)}(G) := \begin{cases} u & \text{if } (u, v) \in E \\ 0 & \text{otherwise} \end{cases}$$

and

$$N_d^{(v)}(G) := \begin{cases} w & \text{if } (v, w) \in E \\ 0 & \text{otherwise} \end{cases}$$

where $G = (V, E)$ is a potential trade partner constellation graph or an actual trade graph. $N_o^{(v)}(G)$ (resp., $N_d^{(v)}(G)$) allows an easy access to the offerer (resp., demander) of P_v. Furthermore, we define

$$N_{o,d}^{(v)}(G) := (N_o^{(v)}(G), N_d^{(v)}(G)).$$

$N_{o,d}^{(v)}(G) = (0, 0)$ indicates that P_v is excluded from the underlying actual trade (or potential trade partner constellation). In Figure 1, for example, $N_{o,d}^{(2)}(G_1^{\text{PTPC}}) = (1, 3)$ while $N_{o,d}^{(4)}(G_1^{\text{PTPC}}) = (0, 0)$. In the following, we write $N_o^{(v)}$, $N_d^{(v)}$, and $N_{o,d}^{(v)}$ whenever the corresponding graph G is apparent from the context.

4.2 Intuition

In the following, we provide an intuition for our novel bartering protocol which is secure against malicious adversaries and allows the determining of an actual trade considering a publicly-known set of trade partner constellations[4] in combination with a given set of parties and their private quotes while maximizing a welfare function that the parties agreed upon prior to executing the protocol.

[4] The trade partner constellation set may include all possible constellations for a fixed set of parties—or for matters of efficiency may be restricted to a subset thereof, e.g., to only include trade cycles of a specific length or to only include constellations in which specific parties get to trade (see the example in Figure 1).

Upon the conclusion of the protocol, a party only learns its *local view* of the determined actual trade (and what can be derived from it in combination with its private input), i.e., its own trade partners and the actual commodities and quantities to be traded with them.

In [21, 22], a bartering protocol is proposed which achieves the same objective but exhibits security only against semi-honest adversaries. The protocol includes two main steps. First, based on the private quotes of the parties, the set of potential trade partner constellations is determined as those trade partner constellations from the given set of trade partner constellations which correspond to a trade all parties would be satisfied with. Then, one potential trade partner constellation is selected from amongst those who maximize the welfare function. This constellation already includes the information as to which party will send (resp., receive) which commodity to (resp., from) which other party as part of the actual trade. In a second step, each party engages in a two-party protocol with each one of its trade partners in order to determine the actual quantities at which the commodities are to be exchanged.

The approach of executing a multi-party protocol followed by a series of two-party protocols is not viable in the context of malicious adversaries as it is not possible to assure that parties provide consistent input for all protocols they participate in. Instead, in order to provide security against malicious adversaries, it is necessary for all steps to be properly intertwined. Intuitively speaking, the novel bartering protocol $\pi_{\mathrm{MB}}^{(\mathcal{W},\mathcal{D})}$ (detailed in Section 6) achieves this as follows: First, given their private quotes, the parties jointly determine an encrypted adjacency matrix representing the compatibility graph. Then, the parties jointly determine a corresponding encrypted matrix where an entry represents the quantity at which one of the respective parties would send the corresponding commodity to the other respective party based on the determined actual trade graph. Specifically, a quantity of zero indicates that the respective edge is not part of the actual trade graph, i.e., the party does not send anything to the other party. In the following, we propose a new gate $\rho_{\text{RSI-}\mathcal{D}}^{\omega_\Delta}$ which allows the selection of the quantities for a given arbitrary discrete probability distribution. At its core, this new gate uses an intricate combination of the known element selection and random bitwise value generation gates in order to implement the sampling according to the prescribed distribution. In a third phase, the parties jointly determine an encrypted indicator vector where the $i-th$ entry of the vector indicates whether the $i-th$ trade partner constellation graph of the given publicly known set of trade partner constellation graphs is a subgraph of the private compatibility graph (defined by their private inputs) in an oblivious fashion. For a given trade partner constellation graph, this step can be implemented by having the parties obliviously multiply together the respective entries of the adjacency matrix—where a resulting encryption of zero corresponds to the fact that this trade partner constellation graph is not a subgraph of the compatibility graph.[5] Subsequently, all parties then jointly generate one encrypted vector for each

[5] To simplify the intuition, we defer the use of the welfare function to prioritize certain trade partner constellation graphs to Section 6.

party. The j-th entry of such a vector encodes the trade partners of this party w.r.t. the j-th trade partner constellation graph as well as the corresponding quantities as determined earlier. In a final step, the protocol selects an actual trade graph uniformly at random from amongst those trade constellation graphs that were previously determined to be a subgraph of the compatibility graph. In the following, we devise a novel multi-party conditional random selection gate (see Section 5.2) to enable this operation. At its core, this new gate $\rho^{i^*}_{CRS}$ carries out a sequence of oblivious shuffles followed by bubble-sort like oblivious comparison and swap operations—upon input of the indicator vector as well as the encrypted vectors that were determined for each party. Eventually, the output of the conditional random selection gate is decrypted in such a fashion that each party only learns its local view of the selected actual trade. In the Appendix, we provide a detailed example which shows the specific workings of the novel protocol.

5 Gates

In this section, we review existing gates which have been integrated into the CDN framework (Section 5.1) and present new gates for privacy-preserving conditional random selection (Section 5.2) and for securely sampling random numbers from a private interval according to an arbitrary discrete distribution $\mathcal{D}$ (Section 5.3). All these gates are used for the implementation of our novel bartering protocol. In the following, we first provide the gate functionality $\mathcal{G}$ before presenting a gate ρ implementing $\mathcal{G}$. The broadcast and round complexities of the presented gates are provided in Table 2. All protocols in this section provide security against malicious adversaries.

5.1 Existing Gates

Definition 7. ($\mathcal{G}_{\mathrm{BR}}$: Secure Computation of the Encrypted Binary Representation of a Paillier Ciphertext [15].) *Let all parties P_v ($v \in \mathscr{P}$) hold $[\![x]\!]$. Then, gate functionality $\mathcal{G}_{BR}$ is given by $(([\![x_0]\!], \ldots, [\![x_m]\!])) \leftarrow \mathcal{G}_{BR}([\![x]\!])$ where $x_{bin} := (x_0, \ldots, x_m)$ is the bit representation of x with $0 \leq x < 2^m$.*

Definition 8. ($\mathcal{G}_{\mathrm{GT}}$: Secure Greater Than Comparison [8].) *Let all parties P_v ($v \in \mathscr{P}$) hold $[\![x_{bin}]\!] := ([\![x_0]\!], \ldots, [\![x_m]\!])$ and $[\![y_{bin}]\!] := ([\![y_0]\!], \ldots, [\![y_m]\!])$ where x_{bin} with $0 \leq x < 2^m$ (resp., y_{bin}) is the binary representation of x (resp., y). Then, gate functionality $\mathcal{G}_{GT}$ is given by $([\![o]\!]) \leftarrow \mathcal{G}_{GT}(([\![x_{bin}]\!], [\![y_{bin}]\!]))$, where $o = [x > y]$.*

Given gate ρ_{GT}, it is straight-forward to derive the corresponding less than (LT), less than or equal (LTE), and greater than or equal (GTE) variants as well as a gate for private equality testing (EQ) (see, e.g., [19]).

Definition 9. ($\mathcal{G}_{\mathrm{RBVG}}$: Random Bitwise Value Generation [10].) *Let all parties P_v ($v \in \mathscr{P}$) hold a public value a such that $2^{l_a - 1} < a \leq 2^{l_a}$. Then,*

Gate	Section	Broadcast Complexity	Round Complexity
ρ_{Mult}	3.4	$O(\iota s)$	$O(1)$
$\rho_{\text{UFI-Mult}}$	5.1	$O(n\iota s)$	$O(n)$
ρ_{BR}	5.1	$O(\iota s^2)$	$O(\iota)$
ρ_{GT}	5.1	$O(\iota s^2)$	$O(log(s))$
ρ_{RBVG}	5.1	$O(\iota s l_a)$	$O(\iota)$
ρ_{ES}	5.1	$O(n\iota s)$	$O(n)$
$\rho^{i^*}_{CRS}$	5.2	$O(n\iota s^2 + mn\iota s)$	$O(n(\iota + log(s)) + mn)$
$\rho^{\omega_\Delta}_{\text{RSI-}\mathcal{D}}$	5.3	$O(\omega_\Delta \iota s(\lvert\mathcal{R}_{\omega_\Delta}\rvert + log(\lvert\mathcal{R}_{\omega_\Delta}\rvert)))$	$O(\omega_\Delta(\iota + \lvert\mathcal{R}_{\omega_\Delta}\rvert))$

Table 2: Gate complexities.

gate functionality $\mathcal{G}_{RBVG}$ *is given by* $((\llbracket r_0 \rrbracket, \ldots, \llbracket r_{l_a-1} \rrbracket)) \leftarrow \mathcal{G}_{RBVG}(a)$ *where* $r_{bin} := (r_0, \ldots, r_{l_a-1})$ *is the bit representation of* r *with* $r \leftarrow_{\$} [0, a)$.

Definition 10. ($\mathcal{G}_{\text{ES}}$: Element Selection [23].) *Let* $N = p \cdot q$ *be the Paillier modulus and let all parties* P_v $(v \in \mathscr{P})$ *hold encryptions* $\llbracket y_1 \rrbracket, \ldots, \llbracket y_n \rrbracket$ *and* $\llbracket w \rrbracket$ *with* $n < min(p, q)$ *and* $w \in \mathbb{N}_n \subset \mathbb{P}$. *Then, gate functionality* $\mathcal{G}_{ES}$ *is given by* $(\llbracket y_w \rrbracket) \leftarrow \mathcal{G}_{ES}((\llbracket y_1 \rrbracket, \ldots, \llbracket y_n \rrbracket, \llbracket w \rrbracket))$.

In [15], Schoenmakers et al. devise a gate ρ_{BR} implementing $\mathcal{G}_{\text{BR}}$. Gates implementing $\mathcal{G}_{\text{GT}}$ and $\mathcal{G}_{\text{RBVG}}$ have been proposed in [8] and [10], respectively. We write $\rho^{\text{int}}_{\text{RBVG}}$ to refer to the variant of gate ρ_{RBVG} which provides output $\llbracket r \rrbracket$ instead of $(\llbracket r_0 \rrbracket, \ldots, \llbracket r_{l_a-1} \rrbracket)$ (cf. Definition 9). A gate implementing gate functionality $\mathcal{G}_{\text{ES}}$ has been introduced recently in [23].

Analogously to [23], we define $(\llbracket x_1 \cdot x_2 \cdots x_n \rrbracket) \leftarrow \rho_{\text{UFI-Mult}}((\llbracket x_1 \rrbracket, \ldots, \llbracket x_n \rrbracket))$ (Unbound Fan In Multiplication) by subsequently executing gate ρ_{Mult} $n - 1$ times (see Section 3.4).

5.2 Privacy-Preserving Conditional Random Selection

The primitive of privacy-preserving conditional random selection has first been introduced in [20]. Given a private set of data records, CRS allows for the random selection of one of the data records that satisfy a specified condition without leaking any information on any data entry. Since the gates from [20] only provide security in the semi-honest model, we can not use them in the context of this paper. Consequently, we have to design a new gate implementing the primitive of conditional random selection with security in the malicious model.

As in [20], our CRS gate operates on encrypted value vectors and an associated encrypted indicator vector. The entries of these vectors at a specific index represent a data record; multiple data records constitute a data table. A binary indicator vector can be used to indicate whether or not the data records satisfy a specified condition. As discussed in [20], by using an integer valued indicator vector, CRS allows for a prioritization of data records. Accordingly, an indicator

vector entry indicates whether the corresponding data record satisfies the specified condition (if greater than zero) and indicates its priority (the higher the value, the higher the priority of the data record).

Our CRS gate makes use of a *mix-net* which allows multiple parties to shuffle a list of (encrypted) elements such that no party knows the permutation linking the input list and the output list. One simple approach to realize a mix-net is to let the parties successively shuffle the target list. A single shuffle operation on a list of ciphertexts $[\![l_1]\!], \ldots, [\![l_n]\!]$ encrypted with a semantically secure homomorphic cryptosystem allowing for ciphertext re-randomization can be realized by choosing a random permutation σ of $\{1, \ldots, n\}$ and computing $Rnd([\![l_{\sigma(1)}]\!]), \ldots, Rnd([\![l_{\sigma(n)}]\!])$. In the malicious model, a shuffle operation performed by a single party has to include a zero-knowledge proof allowing other parties to check the correctness. In the following, we assume the efficient proof system of [9] allowing the implementing of a verifiable secret shuffling operation for ciphertexts. In the following, we refer to the proof of correct shuffling as POCS. The resulting mix-net (consisting of the staggered shuffling operations together with the corresponding POCSs) ensures that if there is at least one honest party choosing a random permutation, it is impossible to link any encrypted element of the input list to any encrypted element of the output list. Furthermore, the proof system from [9] allows a party to prove that several lists of ciphertexts have been shuffled with the same permutation.

Definition 11. ($\mathcal{G}_{CRS}^{i^*}$: Conditional Random Selection). *Let* P_v $(\forall v \in \mathscr{P})$ *hold* m *vectors* $[\![L_i]\!] := ([\![l_{i,1}]\!], \ldots, [\![l_{i,n}]\!])$ *of length* n *where* $i \in \mathbb{N}_m$. *Let* $[\![L_{i^*}]\!]$ $(i^* \in \mathbb{N}_m)$ *be an encrypted indicator vector and* $\{[\![L_1]\!], \ldots, [\![L_m]\!]\} \backslash [\![L_{i^*}]\!]$ *be value vectors. Then, functionality* $\mathcal{G}_{CRS}^{i^*}$ *is given by* $(([\![l_1^*]\!], \ldots, [\![l_m^*]\!])) \leftarrow \mathcal{G}_{CRS}^{i^*}(([\![L_1]\!], \ldots, [\![L_m]\!]))$ *with* $l_i^* := Rnd([\![l_{i,j^*}]\!])$ $(\forall i \in \mathbb{N}_m)$ *where* $j^* \leftarrow_{\$} \{j \in \mathbb{N}_n : l_{i^*,j} = max(l_{i^*,1}, \ldots, l_{i^*,n})\}$.

A gate $\rho_{CRS}^{i^*}$ implementing gate functionality $\mathcal{G}_{CRS}^{i^*}$ is given by Gate 1. Successively, all parties verifyably shuffle the entries of the encrypted input vectors. Party P_1 begins by shuffling $[\![L_i^{(0)}]\!] := [\![L_i]\!]$ $(\forall i \in \mathbb{N}_m)$ in a verifiable fashion resulting in $([\![L_1^{(1)}]\!], \ldots, [\![L_m^{(1)}]\!])$ which P_1 sends to the next party P_2. Note that a party shuffles each encrypted vector using the same permutation in order to maintain the data records. At the end of Step 2 (Gate 1), after P_ι verifiably shuffled $[\![L_i^{(\iota-1)}]\!]$ $(\forall i \in \mathbb{N}_m)$ each party holds $([\![L_1']\!], \ldots, [\![L_m']\!]) := ([\![L_1^{(\iota)}]\!], \ldots, [\![L_m^{(\iota)}]\!])$. Note, that the successive shuffling is a necessary step in $\rho_{CRS}^{i^*}$ which decouples the gate input from the gate output, i.e., it is neither possible for a single party nor for any set of less than ι corrupted parties to link an encrypted data record of the input to the encrypted output data record. The for-loop in Step 4 (Gate 1) operates on two adjacent entries $l'_{i^*,j'-1}$ and $l'_{i^*,j'}$ $(\forall j' \in \{2, \ldots, n\})$ of the encrypted indicator vector $[\![L'_{i^*}]\!] := ([\![l'_{i^*,1}]\!], \ldots, [\![l'_{i^*,n}]\!])$ in each iteration. First, the encrypted bit representations of both entries are computed (Steps 4.1 and 4.2) which constitute the input for gate ρ_{GT} for obliviously determining $[\![o]\!] := [\![[l'_{i^*,j'-1} > l'_{i^*,j'}]]\!]$. Ciphertext $[\![o]\!]$ is used in the inner for-loop (Step 4.4) which iterates over all encrypted vectors $[\![L_1']\!], \ldots, [\![L_m']\!]$ and obliviously swaps

Gate 1. $\rho^{i^*}_{CRS}$ for conditional random selection ($v \in \mathscr{P}$).

Input: P_v holds m vectors $([\![L_1]\!], \ldots, [\![L_m]\!])$
Output: P_v outputs $([\![l^*_1]\!], \ldots, [\![l^*_m]\!])$
1 Set $([\![L^{(0)}_1]\!], \ldots, [\![L^{(0)}_m]\!]) := ([\![L_1]\!], \ldots, [\![L_m]\!])$
2 For v from 1 to ι:
  2.1 P_v verifiably shuffles $[\![L^{(v-1)}_1]\!], \ldots, [\![L^{(v-1)}_m]\!]$ with the same random permutation resulting in $[\![L^{(v)}_1]\!], \ldots, [\![L^{(v)}_m]\!]$ and provides POCS
3 Set $([\![L'_1]\!], \ldots, [\![L'_m]\!]) := ([\![L^{(\iota)}_1]\!], \ldots, [\![L^{(\iota)}_m]\!])$
4 For j' from 2 to n:
  4.1 $([\![(l'_{i^*,j'-1})_{bin}]\!]) \leftarrow \rho_{\mathrm{BR}}([\![(l'_{i^*,j'-1})]\!])$
  4.2 $([\![(l'_{i^*,j'})_{bin}]\!]) \leftarrow \rho_{\mathrm{BR}}([\![(l'_{i^*,j'})]\!])$
  4.3 $([\![o]\!]) \leftarrow \rho_{\mathrm{GT}}(([\![(l'_{i^*,j'-1})_{bin}]\!], [\![(l'_{i^*,j'})_{bin}]\!]))$
  4.4 For i from 1 to m:
    4.4.1 $[\![L'_i[j'-1]]\!] := \rho_{\mathrm{Mult}}(([\![1-o]\!], [\![l'_{i,j'-1}]\!])) +_h \rho_{\mathrm{Mult}}(([\![o]\!], [\![l'_{i,j'}]\!]))$
    4.4.2 $[\![L'_i[j']]\!] := \rho_{\mathrm{Mult}}(([\![1-o]\!], [\![l'_{i,j'}]\!])) +_h \rho_{\mathrm{Mult}}(([\![o]\!], [\![l'_{i,j'-1}]\!]))$
5 Set $([\![l^*_1]\!], \ldots, [\![l^*_m]\!]) := ([\![L'_1[n]]\!], \ldots, [\![L'_m[n]]\!])$

entries $[\![l'_{i,j'-1}]\!]$ and $[\![l'_{i,j'}]\!]$ ($\forall i \in \mathbb{N}_m$) for the case that $o = 1$ (Steps 4.4.1 and 4.4.2). The purpose of the nested for-loop of Step 4 is to iterate over all data records in a bubble-sort fashion in order to propagate a random data record with the maximum indicator vector entry to the rightmost index of the data table, i.e., in Step 5 (Gate 1), $([\![L'_1[n]]\!], \ldots, [\![L'_m[n]]\!])$ contains the selected data record which constitutes the output of the gate.

The broadcast and round complexities of gate $\rho^{i^*}_{CRS}$ are dominated by the nested for-loop (Step 4). Gate ρ_{BR} is called $2(n-1)$ times, gate ρ_{GT} is called $n-1$ times, and gate ρ_{Mult} is called $4(n-1)m$ times. Using the concrete implementations of ρ_{BR}, ρ_{GT}, and ρ_{Mult} mentioned above, the broadcast complexity of $\rho^{i^*}_{CRS}$ is in $O(n\iota s^2 + mn\iota s)$ and the round complexity is in $O(n(\iota + log(s)) + mn)$.

Considering that a verifiable shuffle requires a zero knowledge proof allowing each party to check whether the shuffling operation has been performed correctly, each party deviating from the gate specification is detected. The correctness of gate $\rho^{i^*}_{CRS}$ can be proven analogously to the CRS gates from [20].

Theorem 2. *Let $L^{\langle 0\rangle}_i$, $L^{\langle 1\rangle}_i$, and $[\![b]\!]$ ($b \in \{0,1\}$) with $L^{\langle b\rangle}_i := (l^{\langle b\rangle}_{i,1}, \ldots, l^{\langle b\rangle}_{i,n})$ be given to the simulator ($\forall i \in \mathbb{N}_m$). For input $[\![\tilde{L}_i]\!] := ([\![\tilde{l}_{i,1}]\!], \ldots, [\![\tilde{l}_{i,n}]\!])$ with $[\![\tilde{l}_{i,j}]\!] := [\![(1-b)l^{\langle 0\rangle}_{i,j} + bl^{\langle 1\rangle}_{i,j}]\!]$ ($\forall i \in \mathbb{N}_m, \forall j \in \mathbb{N}_n$), gate $\rho^{i^*}_{CRS}$ can be simulated such that the simulated view is statistically indistinguishable from the real view.*

Proof. W.l.o.g. assume that $P_1, \ldots, P_t$ are corrupted. First, the successive verifiable shuffles (Step 2) have to be simulated which can be done similarly to [8]. The simulator lets P_v for v from 1 to t subsequently verifiably shuffle $([\![\tilde{L}^{(v-1)}_1]\!], \ldots, [\![\tilde{L}^{(v-1)}_m]\!])$ and obtains updated lists $[\![\tilde{L}^{(v)}_1]\!], \ldots, [\![\tilde{L}^{(v)}_m]\!]$. From the respective zero knowledge proof, the simulator extracts the corresponding permutations which

link $([\![\tilde{L}_1]\!], \ldots, [\![\tilde{L}_m]\!])$ and $([\![\tilde{L}_1^{(t)}]\!], \ldots, [\![\tilde{L}_m^{(t)}]\!])$. For the honest parties $P_{t+1}, \ldots, P_\iota$, the simulator chooses random permutations $\sigma_{v'}^{\langle 0\rangle}$ and $\sigma_{v'}^{\langle 1\rangle}$ $(\forall v' \in \{t+1, \ldots, \iota\})$. Up to this point, the simulator thus holds permutations $\sigma_1, \ldots, \sigma_t$ (for the corrupted parties) and permutations $(\sigma_{t+1}^{\langle 0\rangle}, \sigma_{t+1}^{\langle 1\rangle}), \ldots, (\sigma_\iota^{\langle 0\rangle}, \sigma_\iota^{\langle 1\rangle})$ (for the honest parties). What remains to be done is to simulate the zero knowledge proofs for the last $\iota - t$ verifiable shufflings. Let $L_i^{(v')\langle b\rangle} := \sigma_{v'}^{\langle b\rangle}(L_i^{(v'-1)\langle b\rangle})$ $(\forall b, i, v' : b \in \{0,1\}, i \in \mathbb{N}_m, v' \in \{t+1, \iota\})$, let $[\![\tilde{l}_{i,j}^{(v')}]\!] := [\![(1-b)l_{i,j}^{(v')\langle 0\rangle} + bl_{i,j}^{(v')\langle 1\rangle}]\!]$, and let $[\![\tilde{L}_i^{(v')}]\!] := ([\![\tilde{l}_{i,1}^{(v')}]\!], \ldots, [\![\tilde{l}_{i,n}^{(v')}]\!])$. For v' from $t+1$ to ι, the simulator calls the simulator for the zero knowledge proof of the shuffle on inputs $([\![\tilde{L}_1^{(v'-1)}]\!], \ldots, [\![\tilde{L}_m^{(v'-1)}]\!])$ and $([\![\tilde{L}_1^{(v')}]\!], \ldots, [\![\tilde{L}_m^{(v')}]\!])$.

The remaining proof is straight-forward. Next, the simulator sets $([\![\tilde{L}'_1]\!], \ldots, [\![\tilde{L}'_m]\!]) := ([\![\tilde{L}_1^{(\iota)}]\!], \ldots, [\![\tilde{L}_m^{(\iota)}]\!])$. In order to simulate Step 4.1, the simulator of ρ_{BR} is called on input $l'^{\langle 0\rangle}_{i^*,j'-1}$, $l'^{\langle 1\rangle}_{i^*,j'-1}$, and $[\![b]\!]$. The output ρ_{BR} is computed as $[\![(\tilde{l}'_{i^*,j'-1})_{bin}]\!] := [\![(1-b)(l'^{\langle 0\rangle}_{i^*,j'-1})_{bin} + b(l'^{\langle 1\rangle}_{i^*,j'-1})_{bin}]\!]$. Step 4.2 can be simulated analogously. The call of gate ρ_{GT} can be simulated by calling the corresponding simulator on inputs $(l'^{\langle 0\rangle}_{i^*,j'-1})_{bin}$, $(l'^{\langle 1\rangle}_{i^*,j'-1})_{bin}$, $(l'^{\langle 0\rangle}_{i^*,j'})_{bin}$, $(l'^{\langle 1\rangle}_{i^*,j'})_{bin}$, and $[\![b]\!]$. The corresponding output is computed as $[\![\tilde{o}]\!] := [\![(1-b)o^{\langle 0\rangle} + bo^{\langle 1\rangle}]\!]$ where $o^{\langle b\rangle} := [l'^{\langle b\rangle}_{i^*,j'-1} > l'^{\langle b\rangle}_{i^*,j'}]$. Furthermore, the four calls of the multiplication gate ρ_{Mult} in Step 4.4.1 and Step 4.4.2 have to be simulated. The first call of ρ_{Mult} on input $([\![1-o]\!], [\![l'_{i,j'-1}]\!])$ can be simulated by providing the corresponding simulator with input $([\![1]\!] -_h [\![\tilde{o}]\!], [\![\tilde{l}'_{i,j'-1}]\!])$ and output $[\![(1-b)(1-o^{\langle 0\rangle})l'^{\langle 0\rangle}_{i,j'-1} + b(1-o^{\langle 1\rangle})l'^{\langle 1\rangle}_{i,j'-1}]\!]$. The remaining calls of ρ_{Mult} can be simulated analogously. After updating $[\![l'^{\langle b\rangle}_{i,j'-1}]\!] := [\![(1-o^{\langle b\rangle}) \cdot l'^{\langle b\rangle}_{i,j'-1} + o^{\langle b\rangle} \cdot l'^{\langle b\rangle}_{i,j'}]\!]$ $(b \in \{0,1\})$, $[\![\tilde{L}_i[j'-1]]\!]$ is computed as $[\![(1-b)l'^{\langle 0\rangle}_{i,j'-1} + bl'^{\langle 1\rangle}_{i,j'-1}]\!]$. $[\![\tilde{L}_i[j']]\!]$ can be computed analogously. Finally, the simulator sets $[\![\tilde{l}_i^*]\!] := [\![\tilde{l}'_{i,n}]\!]$ $(\forall i \in \mathbb{N}_m)$.

5.3 Secure Random Sampling

Next, we devise a novel gate which allows the random sampling of values from a private positive integer interval $I = [\lambda, \mu]$ with interval width $\Delta := \mu - \lambda$ that are distributed according to a (arbitrary) given discrete probability distribution $\mathcal{D}$ defined by a *probability mass function* $f_{\mathcal{D},I}(l;\Delta) = Pr(x = I[l+1])$ $(l \in \mathbb{N}_\Delta^0)$ where $I[l+1]$ refers to the $(l+1)$-st smallest element in I w.r.t. $\leq$. Note that $\mathcal{D}$ describes the distribution of the elements in I relatively to the interval bounds of I. In our bartering protocol this gate is used to enable the selection of the quantities of the commodities to be traded. Definition 12 captures the desired functionality.

Definition 12. ($\mathcal{G}_{\text{RSI-}\mathcal{D}}^{\omega_\Delta}$: Random Sampling from a Private Interval According to Discrete Distribution $\mathcal{D}$) *Let all parties P_v $(v \in \mathscr{P})$ hold the encrypted interval bounds $[\![\lambda]\!]$, $[\![\mu]\!]$ of an unknown integer interval $[\lambda, \mu]$ with $0 \leq |[\lambda, \mu]| \leq \omega_\Delta$.*

Gate 2. $\rho^{\omega_\Delta}_{\text{RSI-}\mathcal{D}}$ for randomly sampling from a private interval according to distribution $\mathcal{D}$ ($v \in \mathscr{P}$).

Input: P_v holds $[\![\lambda]\!]$, $[\![\mu]\!]$, ω_Δ, $[\![\mathcal{R}_0]\!], \ldots, [\![\mathcal{R}_{\omega_\Delta}]\!]$
Output: P_v outputs $[\![e]\!]$
1 $[\![\Delta]\!] := [\![\mu]\!] -_h [\![\lambda]\!]$
2 For i from 0 to ω_Δ:
2.1 $([\![x_i]\!]) \leftarrow \rho^{\text{int}}_{\text{RBVG}}(|\mathcal{R}_i|)$
2.2 $([\![y_i]\!]) \leftarrow \rho_{\text{ES}}(([\![r_{i,0}]\!], \ldots, [\![r_{i,|\mathcal{R}_i|-1}]\!], [\![x_i]\!]))$
3 $([\![e_\Delta]\!]) \leftarrow \rho_{\text{ES}}(([\![y_0]\!], \ldots, [\![y_{\omega_\Delta}]\!], [\![\Delta]\!]))$
4 $[\![e]\!] := [\![e_\Delta]\!] +_h [\![\lambda]\!]$

Then, functionality $\mathcal{G}^{\omega_\Delta}_{RSI\text{-}\mathcal{D}}$ is given by $([\![e]\!]) \leftarrow \mathcal{G}^{\omega_\Delta}_{RSI\text{-}\mathcal{D}}(([\![\lambda]\!], [\![\mu]\!]))$, where $[\![e]\!]$ is such that $e \leftarrow_{\mathcal{D}} [\lambda, \mu]$ and $\mathcal{D}$, ω_Δ are publicly known.

Gate $\rho^{\omega_\Delta}_{\text{RSI-}\mathcal{D}}$ implementing $\mathcal{G}^{\omega_\Delta}_{\text{RSI-}\mathcal{D}}$ is presented in Gate 2. The problem of computing $[\![e]\!]$ with $e \leftarrow_{\mathcal{D}} [\lambda, \mu]$ is reduced to the problem of computing an element $e_\Delta \leftarrow_{\mathcal{D}} [0, \Delta]$ with $\Delta := \mu - \lambda$. Then, $[\![e]\!]$ is obtained by computing $[\![e_\Delta]\!] +_h [\![\lambda]\!]$. The key idea is to generate $\omega_\Delta + 1$ encrypted lists $[\![\mathcal{R}_0]\!], \ldots, [\![\mathcal{R}_{\omega_\Delta}]\!]$ with

$$\begin{aligned}[\![\mathcal{R}_i]\!] &:= ([\![r_{i,0}]\!], [\![r_{i,1}]\!] \ldots, [\![r_{i,|\mathcal{R}_i|-1}]\!]) \\ &= ([\![0]\!], \overset{j_0}{\ldots}, [\![0]\!], [\![1]\!], \overset{j_1}{\ldots}, [\![1]\!], \ldots, [\![i]\!], \overset{j_i}{\ldots}, [\![i]\!]),\end{aligned}$$

where $i \in \mathbb{N}^0_{\omega_\Delta}$ and $j_l, |\mathcal{R}_i|$ ($l \in \mathbb{N}^0_i$) s.t. $j_l/|\mathcal{R}_i| = f_{\mathcal{D}}(l; i)$. For each of these encrypted lists the parties obliviously select a single entry by calling $([\![x_i]\!]) \leftarrow \rho^{\text{int}}_{\text{RBVG}}(|\mathcal{R}_i|)$. A subsequent call of ρ_{ES} on input $([\![r_{i,0}]\!], \ldots, [\![r_{i,|\mathcal{R}_i|-1}]\!], [\![x_i]\!])$ allows the parties to obliviously obtain the encrypted x_i-th entry of $\mathcal{R}_i$. The resulting ciphertexts $[\![y_0]\!], \ldots, [\![y_{\omega_\Delta}]\!]$ are such that $y_i \leftarrow_{\$} \mathcal{R}_i$ and thus $y_i \leftarrow_{\mathcal{D}} [0, i]$. Next, the parties jointly call gate ρ_{ES} on input $([\![y_0]\!], \ldots, [\![y_{\omega_\Delta}]\!], [\![\Delta]\!])$ and obtain $[\![e_\Delta]\!]$ where $e_\Delta \leftarrow_{\mathcal{D}} [0, \Delta]$. Finally, each party locally computes $[\![e]\!] := [\![\Delta]\!] +_h [\![\lambda]\!]$ which constitutes the output of the gate.

Assuming that for a given gate the lists $\mathcal{R}_0, \ldots, \mathcal{R}_{\omega_\Delta}$ are fixed, $[\![\mathcal{R}_0]\!], \ldots, [\![\mathcal{R}_{\omega_\Delta}]\!]$ can be considered to be publicly known pre-computed gate input. In the following, we provide an example for gate $\rho^{\omega_\Delta}_{\text{RSI-}\mathcal{D}}$ with private input interval $[3, 5]$ where $\mathcal{D}$ is instantiated with the *binomial distribution* $\mathcal{B}$.

Example 1. Let $I = [\lambda, \mu] = [3, 5]$ and let $\omega_\Delta = 4$. Furthermore, let $\mathcal{D}$ be instantiated with the binomial distribution (with success probability 0.5) given by the *probability mass function*

$$f_{\mathcal{B},I}(l; \Delta) = Pr[x = I[l+1]] = \binom{\Delta}{l} 0.5^l (1 - 0,5)^{\Delta - l}$$

with $l \in \mathbb{N}^0_\Delta$. Then, $\mathcal{R}_0, \ldots, \mathcal{R}_4$ with $|\mathcal{R}_i| = 2^i$ ($i \in \mathbb{N}^0_4$) are as specified in Table 3. After determining $[\![\Delta]\!] := [\![2]\!]$, in Step 2 of Gate 2, $[\![y_0]\!], \ldots, [\![y_4]\!]$ are

i	$f(l;i)$ $(\forall l \in [0,i])$	$\mathcal{R}_i$
0	$Pr[x{=}0] = 1$	$\mathcal{R}_0 = (0)$
1	$Pr[x{=}0] = Pr[x{=}1] = 1/2$	$\mathcal{R}_1 = (0,1)$
2	$Pr[x{=}0] = Pr[x{=}2] = 1/4; Pr[x{=}1] = 1/2$	$\mathcal{R}_2 = (0,1,1,2)$
3	$Pr[x{=}0] = Pr[x{=}3] = 1/8; Pr[x{=}1] = Pr[x{=}2] = 3/8$	$\mathcal{R}_3 = (0,1,1,1,2,2,2,3)$
4	$Pr[x{=}0] = Pr[x{=}4] = 1/16; Pr[x{=}1] = Pr[x{=}3] = 1/4;$ $Pr[x{=}2] = 3/8$	$\mathcal{R}_4 = (0,1,1,1,1,2,2,2,2,2,2,3,3,3,3,4)$

Table 3: Construction of $\mathcal{R}_i$ for $\mathcal{D} := \mathcal{B}$ and $\omega_\Delta = 4$ ($i \in \mathbb{N}^0_{\omega_\Delta}$).

jointly computed where $y_i \leftarrow_{\mathcal{B}} [0,i]$. The probabilities for $y_i = l$ ($l \in \mathbb{N}^0_i$) are given in Table 3. In Step 3, the parties obliviously compute $[\![e_\Delta]\!]$ with $e_\Delta := y_\Delta$. Consequently, after locally computing $[\![e]\!] := [\![\Delta]\!] +_h [\![\lambda]\!]$, it holds that $Pr[e{=}3] = Pr[e{=}5] = 1/4$ and $Pr[e{=}4] = 1/2$.

Due to the construction of $\mathcal{R}_0, \ldots, \mathcal{R}_{\omega_\Delta}$ and under the assumption that $0 \leq \mu - \lambda \leq \omega_\Delta$, it holds that gate output $[\![e]\!]$ is selected as $e \leftarrow_{\mathcal{D}} [\lambda, \mu]$. The broadcast complexity of $\rho^{\omega_\Delta}_{\text{RSI-}\mathcal{D}}$ is in $O(\omega_\Delta \iota s(|\mathcal{R}_{\omega_\Delta}| + log(|\mathcal{R}_{\omega_\Delta}|)))$ and the round complexity is in $O(\omega_\Delta(\iota + |\mathcal{R}_{\omega_\Delta}|))$.

Theorem 3. *Let ω_Δ, $\mathcal{R}_0, \ldots, \mathcal{R}_{\omega_\Delta}$, $\lambda^{\langle 0\rangle}$, $\lambda^{\langle 1\rangle}$, $\mu^{\langle 0\rangle}$, $\mu^{\langle 1\rangle}$, and $[\![b]\!]$ be given to the simulator. For input $[\![\tilde{\lambda}]\!] := [\![(1-b)\lambda^{\langle 0\rangle} + b\lambda^{\langle 1\rangle}]\!]$ and $[\![\tilde{\mu}]\!] := [\![(1-b)\mu^{\langle 0\rangle} + b\mu^{\langle 1\rangle}]\!]$, gate $\rho^{\omega_\Delta}_{RSI\text{-}\mathcal{D}}$ can be simulated such that the simulated view is statistically indistinguishable from the real view.*

Proof. First, the simulator computes $[\![\tilde{\Delta}]\!] := [\![(1-b)\Delta^{\langle 0\rangle} + b\Delta^{\langle 1\rangle}]\!]$ with $\Delta^{\langle 0\rangle} := \mu^{\langle 0\rangle} - \lambda^{\langle 0\rangle}$ and $\Delta^{\langle 1\rangle} := \mu^{\langle 1\rangle} - \lambda^{\langle 1\rangle}$. The $\omega_\Delta + 1$ calls of $\rho^{\text{int}}_{\text{RBVG}}$ and ρ_{ES} can be simulated by calling the simulator of $\rho^{\text{int}}_{\text{RBVG}}$ on input $[\![b]\!]$, $|\mathcal{R}_i|$, $x_i^{\langle 0\rangle}$, $x_i^{\langle 1\rangle}$ where $x_i^{\langle 0\rangle}, x_i^{\langle 1\rangle} \leftarrow_{\$} [0, |\mathcal{R}_i|)$ and by calling the simulator of ρ_{ES} on input $[\![b]\!]$, $x_i^{\langle 0\rangle}$, $x_i^{\langle 1\rangle}$, $\mathcal{R}_i = (r_{i,0}, r_{i,1}, \ldots, r_{i,|\mathcal{R}_i|-1})$. The output of ρ_{ES} is set to $[\![\tilde{y}_i]\!] := [\![(1-b)y_i^{\langle 0\rangle} + by_i^{\langle 1\rangle}]\!]$ where $y_i^{\langle 0\rangle} := r_{i,x_i^{\langle 0\rangle}}$ and $y_i^{\langle 1\rangle} := r_{i,x_i^{\langle 1\rangle}}$. Step 3 of Gate 2 can be simulated by calling the simulator of ρ_{ES} on inputs $[\![b]\!]$, $\Delta^{\langle 0\rangle}$, $\Delta^{\langle 1\rangle}$, $y_i^{\langle 0\rangle}$, $y_i^{\langle 1\rangle}$ ($\forall i \in \mathbb{N}^0_{\omega_\Delta}$). The output of ρ_{ES} is computed as $[\![\tilde{e}_\Delta]\!] := [\![(1-b)y^{\langle 0\rangle}_{\Delta^{\langle 0\rangle}} + by^{\langle 1\rangle}_{\Delta^{\langle 1\rangle}}]\!]$. Finally, the simulator computes $[\![\tilde{e}]\!] := [\![(1-b)(y^{\langle 0\rangle}_{\Delta^{\langle 0\rangle}} + \lambda^{\langle 0\rangle}) + b(y^{\langle 1\rangle}_{\Delta^{\langle 1\rangle}} + \lambda^{\langle 1\rangle})]\!]$. Since the simulated values are consistent with those of a real gate execution, the simulated view and the real view are statistically indistinguishable.

6 Multi-Party Bartering Protocol

In the following, we introduce our novel privacy-preserving multi-party bartering protocol.

Definition 13. ($\mathcal{F}^{(\mathcal{W},\mathcal{D})}_{\text{MB}}$: Multi-Party Bartering.) *Let party P_v hold its private input $\mathbf{q}^{(v)}$ ($\forall v \in \mathscr{P}$). Furthermore, let $\mathbb{G}^{TPC}$ be a publicly known arbitrary non-empty set of trade partner constellations graphs for ι parties, let $\mathcal{D}$ be a publicly*

known arbitrary discrete distribution, and let $\mathcal{W}(\cdot) : \mathbb{G}^{TPC} \rightarrow \mathbb{N} \cup \{0\}$ *be a publicly known arbitrary welfare function. Then, functionality* $\mathcal{F}_{MB}^{(\mathcal{W},\mathcal{D})}$ *is defined as*

$$\left.\begin{array}{ll}(o_1,\ldots,o_\iota) & \text{if } \mathbb{G}^{PTPC} \neq \emptyset \\ (0) & \text{otherwise}\end{array}\right\} \leftarrow \mathcal{F}_{MB}^{(\mathcal{W},\mathcal{D})}(\mathbf{q}^{(1)},\ldots,\mathbf{q}^{(\iota)},\mathbb{G}^{TPC})$$

with

$$\begin{aligned}&o_v := (N_{o,d}^{(v)}(G^{AT}), q_{c_o^{(N_o^{(v)})}}^{(N_o^{(v)},v)}, q_{c_o^{(v)}}^{(v,N_d^{(v)})}),\\ &G^{AT} \leftarrow_{max_{\mathcal{W}}} \mathbb{G}^{PTPC} \sqsubseteq \mathbb{G}^{TPC}, \text{ and}\\ &q_{c_o^{(v)}}^{(v,N_d^{(v)})} \leftarrow_{\mathcal{D}} Q_o^{(v)} \cap Q_d^{(N_d^{(v)})}.\end{aligned}$$

In an ideal world where a trusted third party exists, functionality $\mathcal{F}_{\text{MB}}^{(\mathcal{W},\mathcal{D})}$ can be computed as follows: Each party P_v ($v \in \mathscr{P}$) sends its private input $\mathbf{q}^{(v)}$ to the trusted third party which additionally is given the public set of trade partner constellation graphs $\mathbb{G}^{\text{TPC}}$, the public welfare function $\mathcal{W}$, and the public discrete distribution $\mathcal{D}$. With the knowledge of $\mathbf{Q} = \{\mathbf{q}^{(1)},\ldots,\mathbf{q}^{(\iota)}\}$, the trusted third party then locally computes $\mathbb{G}^{\text{PTPC}} \subseteq \mathbb{G}^{\text{TPC}}$. In case that $\mathbb{G}^{\text{PTPC}} \neq \emptyset$, the trusted third party selects a random potential trade partner constellation graph $G^{\text{PTPC}} = (V, E)$ with maximum welfare, selects $q_{c_o^{(v)}}^{(v,w)} \leftarrow_{\mathcal{D}} Q_o^{(v)} \cap Q_d^{(w)}$ for each pair of edges $(u,v),(v,w) \in E$, and sends $(N_{o,d}^{(v)}(G^{\text{AT}}), q_{c_o^{(u)}}^{(u,v)}, q_{c_o^{(v)}}^{(v,w)})$ to party P_v. Otherwise, the trusted third party returns 0 to all parties.

In the real world, where no trusted party exists, protocol $\pi_{\text{MB}}^{(\mathcal{W},\mathcal{D})}$ (see Protocol 1) is executed in order to compute functionality $\mathcal{F}_{\text{MB}}^{(\mathcal{W},\mathcal{D})}$. Following the intuition provided in Section 4, $\pi_{\text{MB}}^{(\mathcal{W},\mathcal{D})}$ can be split up into the following phases:

Input Phase: P_v ($\forall v \in \mathscr{P}$) holds private input $\mathbf{q}^{(v)} = ((c_o^{(v)}, \overline{q}_o^{(v)}), (c_d^{(v)}, \underline{q}_d^{(v)}))$. $\pi_{\text{MB}}^{(\mathcal{W},\mathcal{D})}$ is composed of gates expecting encrypted integer inputs as well as of gates expecting encrypted binary inputs. In order to reduce the calls of the expensive ρ_{BR} gate, we require that all parties commit their private input as encrypted integers $[\![\mathbf{q}^{(v)}]\!] := (([\![c_o^{(v)}]\!], [\![\overline{q}_o^{(v)}]\!]), ([\![c_d^{(v)}]\!], [\![\underline{q}_d^{(v)}]\!]))$ as well as encrypted bit decompositions $[\![\mathbf{q}_{\text{bin}}^{(v)}]\!] := (([\![(c_o^{(v)})_{\text{bin}}]\!], [\![(\overline{q}_o^{(v)})_{\text{bin}}]\!]), ([\![(c_d^{(v)})_{\text{bin}}]\!], [\![(\underline{q}_d^{(v)})_{\text{bin}}]\!]))$. Thus, it is not sufficient that all parties just mutually prove plaintext knowledge of their committed input. Instead, P_v has to also prove the consistency of its committed input to guarantee the correctness of the protocol output. More precisely, P_v has to perform the following proofs which can be accomplished as described in [23]:

1.) P_v has to prove that it knows the plaintexts of $[\![c_o^{(v)}]\!], [\![c_d^{(v)}]\!], [\![\overline{q}_o^{(v)}]\!], [\![\underline{q}_d^{(v)}]\!]$.
2.) P_v has to prove that each $[\![(c_o^{(v)})_{\text{bin}}]\!]$, $[\![(c_d^{(v)})_{\text{bin}}]\!], [\![(\overline{q}_o^{(v)})_{\text{bin}}]\!]$, $[\![(\underline{q}_d^{(v)})_{\text{bin}}]\!]$ is a tuple of encrypted bits.

Protocol 1. $\pi_{\mathrm{MB}}^{(\mathcal{W},\mathcal{D})}$ for obliviously selecting an actual trade.

Input: P_v holds private input $\mathbf{q}^{(v)}$ $(v \in \mathscr{P})$

Output: P_v outputs $o_v := \begin{cases} (N_{o,d}^{(v)}(G^{\mathrm{AT}}), q_{c_o^{(N_o^{(v)})}}^{(N_o^{(v)},v)}, q_{c_o^{(v)}}^{(v,N_d^{(v)})}) & \text{if } \mathbb{G}^{\mathrm{PTPC}} \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$

1 Input Phase

1.1 $((\llbracket \mathbf{q}^{(1)} \rrbracket, \llbracket \mathbf{q}_{bin}^{(1)} \rrbracket, \ldots, \llbracket \mathbf{q}^{(\iota)} \rrbracket, \llbracket \mathbf{q}_{bin}^{(\iota)} \rrbracket)) \leftarrow \mathrm{InputPhase}(\mathbf{q}^{(1)}, ..., \mathbf{q}^{(\iota)})$

2 Construction Phase

2.1 For each P_v $(v \in \mathscr{P})$:

2.1.1 For each $P_{v'}$ $(v' \in \mathscr{P} \backslash \{v\})$:

2.1.1.1 $(\llbracket b_1 \rrbracket) \leftarrow \rho_{\mathrm{EQ}}((\llbracket (c_o^{(v)})_{bin} \rrbracket, \llbracket (c_d^{(v')})_{bin} \rrbracket))$

2.1.1.2 $(\llbracket b_2 \rrbracket) \leftarrow \rho_{\mathrm{GTE}}((\llbracket (\overline{q}_o^{(v)})_{bin} \rrbracket, \llbracket (\underline{q}_d^{(v')})_{bin} \rrbracket))$

2.1.1.3 $(\llbracket a_{v,v'} \rrbracket) \leftarrow \rho_{\mathrm{Mult}}((\llbracket b_1 \rrbracket, \llbracket b_2 \rrbracket))$

3 Negotiation Phase

3.1 For each P_v $(v \in \mathscr{P})$:

3.1.1 For each $P_{v'}$ $(v' \in \mathscr{P} \backslash \{v\})$:

3.1.1.1 $(\llbracket \overline{q}_o'^{(v)} \rrbracket) \leftarrow \rho_{\mathrm{Mult}}((\llbracket \overline{q}_o^{(v)} \rrbracket, \llbracket a_{v,v'} \rrbracket))$, $(\llbracket \underline{q}_d'^{(v')} \rrbracket) \leftarrow \rho_{\mathrm{Mult}}((\llbracket \underline{q}_d^{(v')} \rrbracket, \llbracket a_{v,v'} \rrbracket))$

3.1.1.2 $\llbracket a'_{v,v'} \rrbracket := (\llbracket q_{c_o^{(v)}}^{(v,v')} \rrbracket) \leftarrow \rho_{\mathrm{RSI}\text{-}\mathcal{D}}^{\omega_\Delta}((\llbracket \underline{q}_d'^{(v')} \rrbracket, \llbracket \overline{q}_o'^{(v)} \rrbracket))$

3.1.2 $\llbracket a'_{v,0} \rrbracket, \llbracket a'_{0,v} \rrbracket := \llbracket 0 \rrbracket$

4 Evaluation Phase

4.1 For each $G_j^{\mathrm{TPC}} = (V, E) \in \mathbb{G}^{\mathrm{TPC}}$ with $(v_i, v_i') \in E$ and $j \in \mathbb{N}_{|\mathbb{G}^{\mathrm{TPC}}|}, i \in \mathbb{N}_{|E|}$:

4.1.1 $(\llbracket e_j \rrbracket) \leftarrow \rho_{\mathrm{UFI\text{-}Mult}}((\llbracket a_{v_1,v_1'} \rrbracket, \ldots, \llbracket a_{v_{|E|},v_{|E|}'} \rrbracket))$

4.2 Each party sets $\llbracket L \rrbracket := (\llbracket e_1 \rrbracket, \ldots, \llbracket e_{|\mathbb{G}^{\mathrm{TPC}}|} \rrbracket)$

5 Prioritization Phase

5.1 Each party locally computes
$\llbracket L_0 \rrbracket := (\llbracket e_1 \rrbracket \times_h \mathcal{W}(G_1^{\mathrm{TPC}}), \ldots, \llbracket e_{|\mathbb{G}^{\mathrm{TPC}}|} \rrbracket \times_h \mathcal{W}(G_{|\mathbb{G}^{\mathrm{TPC}}|}^{\mathrm{TPC}}))$

6 Mapping Phase

6.1 All parties jointly generate $\llbracket L_1 \rrbracket, \ldots, \llbracket L_\iota \rrbracket$ where

$$\begin{aligned} \llbracket L_1 \rrbracket := (&\llbracket N_{o,d}^{(1)}(G_1^{\mathrm{TPC}}) \rrbracket \,||_h\, \llbracket a'_{N_o^{(1)},1} \rrbracket \,||_h\, \llbracket a'_{1,N_d^{(1)}} \rrbracket, \\ &\ldots, \llbracket N_{o,d}^{(1)}(G_{|\mathbb{G}^{\mathrm{TPC}}|}^{\mathrm{TPC}}) \rrbracket \,||_h\, \llbracket a'_{N_o^{(1)},1} \rrbracket \,||_h\, \llbracket a'_{1,N_d^{(1)}} \rrbracket) \\ &\vdots \\ \llbracket L_\iota \rrbracket := (&\llbracket N_{o,d}^{(\iota)}(G_1^{\mathrm{TPC}}) \rrbracket \,||_h\, \llbracket a'_{N_o^{(\iota)},\iota} \rrbracket \,||_h\, \llbracket a'_{\iota,N_d^{(\iota)}} \rrbracket, \\ &\ldots, \llbracket N_{o,d}^{(\iota)}(G_{|\mathbb{G}^{\mathrm{TPC}}|}^{\mathrm{TPC}}) \rrbracket \,||_h\, \llbracket a'_{N_o^{(\iota)},\iota} \rrbracket \,||_h\, \llbracket a'_{\iota,N_d^{(\iota)}} \rrbracket) \end{aligned}$$

7 Selection Phase

7.1 $((\llbracket l_0^* \rrbracket, \llbracket l_1^* \rrbracket, \ldots, \llbracket l_\iota^* \rrbracket)) \leftarrow \rho_{CRS}^{i^*=0}((\llbracket L_0 \rrbracket, \llbracket L_1 \rrbracket, \ldots, \llbracket L_\iota \rrbracket))$

7.2 $(\llbracket (l_0^*)_{bin} \rrbracket) \leftarrow \rho_{\mathrm{BR}}(\llbracket l_0^* \rrbracket)$

7.3 $(\llbracket b \rrbracket) \leftarrow \rho_{\mathrm{EQ}}((\llbracket (l_0^*)_{bin} \rrbracket, \llbracket 0_{bin} \rrbracket))$

7.4 For v from 1 to ι:

7.4.1 $(\llbracket o_v \rrbracket) \leftarrow \rho_{\mathrm{Mult}}((\llbracket 1 - b \rrbracket, \llbracket l_v^* \rrbracket))$

8 Output Phase

8.1 $(o_1, \ldots, o_\iota) \leftarrow \mathrm{OutputPhase}((\llbracket o_1 \rrbracket, \ldots, \llbracket o_\iota \rrbracket))$

3.) P_v has to prove that $[\![(c_o^{(v)})_{\text{bin}}]\!]$, $[\![(c_d^{(v)})_{\text{bin}}]\!]$, $[\![(\overline{q}_o^{(v)})_{\text{bin}}]\!]$, $[\![(\underline{q}_d^{(v)})_{\text{bin}}]\!]$ are in fact the encrypted bit decompositions of $[\![c_o^{(v)}]\!], [\![c_d^{(v)}]\!], [\![\overline{q}_o^{(v)}]\!], [\![\underline{q}_d^{(v)}]\!]$, respectively.

Construction Phase: Based on their quotes, all parties jointly construct the encrypted adjacency matrix $[\![A]\!] := ([\![a_{v,v'}]\!])_{\iota\times\iota}$ of compatibility graph G^{C} with $a_{v,v'} := [(c_o^{(v)} = c_d^{(v')}) \wedge (\overline{q}_o^{(v)} \geq \underline{q}_d^{(v')})]$ for $v, v' \in \mathscr{P}$, and $v \neq v'$. The ciphertext $[\![[(c_o^{(v)} = c_d^{(v')}) \wedge (\overline{q}_o^{(v)} \geq \underline{q}_d^{(v')})]]\!]$ is computed in three steps (see Steps 2.1.1.1 - 2.1.1.3, Protocol 1).

Negotiation Phase: All parties jointly compute a second encrypted matrix $[\![A']\!] := ([\![a'_{v,v'}]\!])_{\iota\times\iota}$ where $a'_{v,v'}$ $(v \neq v')$ corresponds to the quantity of commodity $c_o^{(v)}$ which P_v has to send to $P_{v'}$ provided that (v, v') is an edge in the selected actual trade graph G^{AT}. In order to determine a matrix entry $[\![a'_{v,v'}]\!]$, the parties jointly execute gate $\rho^{\omega_\Delta}_{\text{RSI-}\mathcal{D}}$ on inputs $\overline{q}_o'^{(v)}$ and $\underline{q}_d'^{(v')}$ which are computed as $\rho_{\text{Mult}}([\![\overline{q}_o^{(v)}]\!], [\![a_{v,v'}]\!])$ and $\rho_{\text{Mult}}([\![\underline{q}_d^{(v')}]\!], [\![a_{v,v'}]\!])$, respectively, in Step 3.1.1.1. The two joint calls of the multiplication gate are necessary in order to prevent that gate $\rho^{\omega_\Delta}_{\text{RSI-}\mathcal{D}}$ is called on invalid input which occurs if $\overline{q}_o^{(v)} < \underline{q}_d^{(v')}$. In this case, $\overline{q}_o'^{(v)} = \underline{q}_d^{(v')} = 0$ and $\rho^{\omega_\Delta}_{\text{RSI-}\mathcal{D}}$ returns a fresh encryption of zero. Note that due to the fact that it is necessary to hide the edges of G^{C}, $\rho^{\omega_\Delta}_{\text{RSI-}\mathcal{D}}$ has to be executed for each pair of parties. In order to cover the case where for a trade partner constellation graph G^{TPC} there are some parties P_v with $N^{(v)}_{o,d}(G^{\text{TPC}}) = (0, 0)$, the parties jointly generate dummy encryptions of zero (see Step 3.1.2) which can be used as a placeholder in the mapping phase. Eventually, these placeholders indicate that for a party without any trading partners there is nothing to exchange. Using gate $\rho^{\omega_\Delta}_{\text{RSI-}\mathcal{D}}$ to determine the quantities of commodities that are to be exchanged w.r.t. the selected actual trade provides for a flexible solution for negotiation since the parties can jointly agree on $\mathcal{D}$ before the protocol is executed. For example, by setting $\mathcal{D}$ to the binomial distribution, the probability of selecting imbalanced quantities (i.e., a party makes a good deal at the expense of another party) can be reduced which can occur in the bartering protocols from [6, 21] where the quantities are selected uniformly at random from the corresponding overlap intervals (for details see [23]).

Evaluation Phase: For each $G_j^{\text{TPC}} = (V, E) \in \mathbb{G}^{\text{TPC}}$ $(j \in \mathbb{N}_{\mathbb{G}^{\text{TPC}}})$, the parties obliviously check whether or not $G_j^{\text{TPC}} \sqsubseteq G^{\text{C}}$ by calling $\rho_{\text{UFI-Mult}}$ on input $([\![a_{v_1,v'_1}]\!], \ldots, [\![a_{v_{|E|},v'_{|E|}}]\!])$ with $(v_i, v'_i) \in E$ and $i \in \mathbb{N}_{|E|}$. Note that $[\![a_{v_1,v'_1}]\!], \ldots, [\![a_{v_{|E|},v'_{|E|}}]\!]$ are taken from the jointly computed adjacency matrix $[\![A]\!]$ of G^{C}. At the end of the evaluation phase, each party holds vector $[\![L]\!] := ([\![e_1]\!], \ldots, [\![e_{|\mathbb{G}^{\text{TPC}}|}]\!])$ where

$$e_j = \begin{cases} 1 & \text{if } G_j^{\text{TPC}} \sqsubseteq G^{\text{C}} \\ 0 & \text{otherwise} \end{cases}.$$

The trade partner constellation graphs G_j^{TPC} with $e_j = 1$ constitute the set of potential trade partner constellation graphs $\mathbb{G}^{\mathrm{PTPC}}$.

Prioritization Phase: Function $\mathcal{W}(\cdot) : \mathbb{G}^{\mathrm{TPC}} \rightarrow \mathbb{N} \cup \{0\}$ maps each trade partner constellation graph to its welfare. Each party locally computes $[\![e_j]\!] \times_h \mathcal{W}(G_j^{\mathrm{TPC}})$ $(\forall j \in \mathbb{N}_{|\mathbb{G}^{\mathrm{TPC}}|})$ resulting in an encrypted vector $[\![L_0]\!] := ([\![e_1 \cdot \mathcal{W}(G_1^{\mathrm{TPC}})]\!], \ldots,$ $[\![e_{|\mathbb{G}^{\mathrm{TPC}}|} \cdot \mathcal{W}(G_{|\mathbb{G}^{\mathrm{TPC}}|}^{\mathrm{TPC}})]\!])$. The prioritization phase ensures that in the selection phase, the potential trade partner constellation graph with the maximum welfare is selected for computing the actual trade graph (see below). Note that a corrupted party deviating from the prescribed computation of $[\![L_0]\!]$ is detected in the selection phase because the encrypted vector will differ from the encrypted vectors computed by other parties.

Mapping Phase: For each party P_v, an encrypted vector

$$[\![L_v]\!] = ([\![N_{o,d}^{(v)}(G_1^{\mathrm{TPC}})]\!] \mid\mid_h [\![a'_{N_o^{(v)},v}]\!] \mid\mid_h [\![a'_{v,N_d^{(v)}}]\!], \ldots,$$
$$[\![N_{o,d}^{(v)}(G_{|\mathbb{G}^{\mathrm{TPC}}|}^{\mathrm{TPC}})]\!] \mid\mid_h [\![a'_{N_o^{(v)},v}]\!] \mid\mid_h [\![a'_{v,N_d^{(v)}}]\!]) \ (\forall v \in \mathscr{P})$$

is jointly computed. The j-th entry of a vector L_v contains the trade partners $N_o^{(v)}(G_j^{\mathrm{TPC}})$ and $N_d^{(v)}(G_j^{\mathrm{TPC}})$ of P_v as well as the quantities to be received and sent in case that G_j^{TPC} is selected as actual trade graph by protocol $\pi_{\mathrm{MB}}^{(\mathcal{W},\mathcal{D})}$. The purpose of the mapping phase is to determine the parties' local views for each trade partner constellation graph.[6]

Selection Phase: To this point, each G_j^{TPC} is associated with an encrypted vector $([\![l_{j,0}]\!], [\![l_{j,1}]\!], \ldots, [\![l_{j,\iota}]\!])$ where $l_{0,j} = 0$ if $G_j^{\mathrm{TPC}} \not\sqsubseteq G^{\mathrm{C}}$ and $l_{0,j} = \mathcal{W}(G_j^{\mathrm{TPC}})$ otherwise. Furthermore, $l_{v,j} = (N_{o,d}^{(v)}(G_j^{\mathrm{TPC}}), a'_{N_o^{(v)},v}, a'_{v,N_d^{(v)}})$ contains P_v's local view of G_j^{TPC} $(\forall v \in \mathscr{P})$. The parties now jointly call $\rho_{CRS}^{i^*=0}$ (see Section 5.2) on the common input $([\![L_0]\!], \ldots, [\![L_\iota]\!])$. In case that $\mathbb{G}^{\mathrm{PTPC}} \neq \emptyset$ (there exists at least one $j \in \mathbb{N}_{\mathbb{G}^{\mathrm{TPC}}}$ such that $l_{0,j} \neq 0$), $\rho_{CRS}^{i^*=0}$ is used to obliviously determine a random actual trade grade G^{AT} which maximizes the welfare function. The first entry of the encrypted output $[\![l_0^*]\!]$ of $\rho_{CRS}^{i^*=0}$ corresponds to the encrypted welfare of G^{AT}. The remaining encrypted entries $[\![l_1^*]\!], \ldots, [\![l_\iota^*]\!]$ correspond to the parties' local views w.r.t. G^{AT}. Otherwise (i.e., $\mathbb{G}^{\mathrm{PTPC}} = \emptyset$), $\rho_{CRS}^{i^*=0}$ is used to select $G^{\mathrm{TPC}} \leftarrow_{\$} \mathbb{G}^{\mathrm{TPC}}$ and $l_0^* = 0$. The remaining steps of the selection phase deal with latter case where $\mathbb{G}^{\mathrm{PTPC}} = \emptyset$. First, the parties obliviously check whether or not $l_0^* = 0$ by jointly calling $([\![b]\!]) \leftarrow \rho_{\mathrm{EQ}}(([\![(l_0^*)_{bin}]\!], [\![0_{bin}]\!]))$ (Step 7.3, Protocol 1). Subsequently, the local view $[\![l_v^*]\!]$ of each party P_v is multiplied with the encrypted result $[\![b]\!]$ of the equality check by jointly calling $([\![o_v]\!]) \leftarrow \rho_{\mathrm{Mult}}(([\![1-b]\!], [\![l_v^*]\!]))$ such that all values l_v^* are obliviously zeroed out

[6] Note that in order to perform the homomorphic concatenation operation, it has to be assured that no 'wrapping' occurs in $\mathbb{P}$ (see Section 3.2). For the given context, however, this issue does not constitute a practical limitation.

for the case that no actual trade graph exists (Step 7.4, Protocol 1).

Output Phase: All parties jointly decrypt $([\![o_1]\!], \ldots, [\![o_\iota]\!])$ such that only P_v learns o_v $(\forall v \in \mathscr{P})$. From o_v a party either learns its local view of the selected G^{AT} (for the case that $\mathbb{G}^{\mathrm{PTPC}} \neq \emptyset$) or otherwise that there does not exist a G^{AT} for the current set of parties in combination with their private input quotes and the given set of trade partner constellation graphs $\mathbb{G}^{\mathrm{TPC}}$.

Let $O(\rho)$ refer to the (broadcast, round) complexity of gate ρ. The broadcast and round complexities of $\pi_{\mathrm{MB}}^{(\mathcal{W},\mathcal{D})}$ are dominated by the computations performed in the construction phase, in the negotiation phase, in the evaluation phase, and in the selection phase. The corresponding complexities are in $O(\iota^2(O(\rho_{\mathrm{EQ}}) + O(\rho_{\mathrm{GTE}}) + O(\rho_{\mathrm{Mult}}))), O(\iota^2(O(\rho_{\mathrm{Mult}}) + O(\rho_{\mathrm{RSI\text{-}}\mathcal{D}}^{\omega_\Delta}))), O(|\mathbb{G}^{\mathrm{TPC}}|O(\rho_{\mathrm{UFI\text{-}Mult}}))$, and $O(O(\rho_{CRS}^{i^*=0}) + \iota O(\rho_{\mathrm{Mult}}))$.

In the following, we show the correctness of $\pi_{\mathrm{MB}}^{(\mathcal{W},\mathcal{D})}$, i.e., that $\pi_{\mathrm{MB}}^{(\mathcal{W},\mathcal{D})}$ always computes the correct output (if at least one party is honest), provided that the protocol is not prematurely aborted. In order to prove the correctness of the protocol, we have to show that (i) it suffices that each party performs proofs 1.) - 3.) as specified for the input phase and that (ii) $\pi_{\mathrm{MB}}^{(\mathcal{W},\mathcal{D})}$ in fact computes a random actual trade graph which maximizes the welfare function based on the parties' input quotes and the publicly known set of trade partner constellation graphs iff the set of potential trade partner constellation graphs is not empty. In [23], (i) has already been proven for the two-party case and can be proven analogously for the multi-party case. Thus, in the following, we focus on showing (ii).

In case that $\mathbb{G}^{\mathrm{PTPC}} = \emptyset$, the result of the evaluation phase is a vector $[\![L]\!] := ([\![e_1]\!], \ldots, [\![e_{|\mathbb{G}^{\mathrm{TPC}}|}]\!])$ with $e_1 = \ldots = e_{|\mathbb{G}^{\mathrm{TPC}}|} = 0$ since there exists no $G_j^{\mathrm{TPC}} \in \mathbb{G}^{\mathrm{TPC}}$ with $G_j^{\mathrm{TPC}} \sqsubseteq G^{\mathrm{C}}$. Furthermore, the prioritization phase does not change any value e_j and thus $[\![L_0]\!] := [\![L]\!]$ $(j \in \mathbb{N}_{|\mathbb{G}^{\mathrm{TPC}}|})$. This implies that gate $\rho_{CRS}^{i^*=0}$ returns an encrypted vector $([\![l_0^*]\!], [\![l_1^*]\!], \ldots, [\![l_\iota^*]\!])$ with $l_0^* = 0$ in the selection phase and in the output phase each party P_v learns its output value $o_v = 0$ which indicates that there exists no actual trade.

In case that $\mathbb{G}^{\mathrm{PTPC}} \neq \emptyset$, the result of the evaluation phase is a vector $[\![L]\!]$ such that L has *Hamming weight* $\mathcal{H}(L) = |\mathbb{G}^{\mathrm{PTPC}}|$. The prioritization phase determines the encrypted vector $[\![L_0]\!] := ([\![l_{0,1}]\!], \ldots, [\![l_{0,|\mathbb{G}^{\mathrm{TPC}}|}]\!])$ where $[\![l_{0,j}]\!]$ is associated with G_j^{TPC} and $l_{0,j} = \mathcal{W}(G_j^{\mathrm{TPC}})$ if $G_j^{\mathrm{TPC}} \sqsubseteq G^{\mathrm{C}}$ and $l_{0,j} = 0$ otherwise. According to the definition of $\mathcal{G}_{CRS}^{i^*}$, gate $\rho_{CRS}^{i^*=0}$ returns an encrypted vector $([\![l_0^*]\!], [\![l_1^*]\!], \ldots, [\![l_\iota^*]\!]) := ([\![l_{0,j^*}]\!], [\![l_{1,j^*}]\!], \ldots, [\![l_{\iota,j^*}]\!])$ where $j^* \leftarrow_{\$} \{j \in \mathbb{N}_{|\mathbb{G}^{\mathrm{TPC}}|} : G_j^{\mathrm{TPC}} = max(\mathcal{W}(G_1^{\mathrm{TPC}}), \ldots, \mathcal{W}(G_{|\mathbb{G}^{\mathrm{TPC}}|}^{\mathrm{TPC}}))\}$. Since in Step 7.3 (Protocol 1) $b = 0$, $[\![o_v]\!] := [\![l_v^*]\!]$ for all $v \in \mathscr{P}$. At the end of the output phase, each party P_v learns its local view of the selected actual trade.

The security of protocol $\pi_{\mathrm{MB}}^{(\mathcal{W},\mathcal{D})}$ directly follows from Theorem 1 (Section 3.4).

Corollary 1. *Protocol $\pi_{MB}^{(\mathcal{W},\mathcal{D})}$ securely computes functionality $\mathcal{F}_{MB}^{(\mathcal{W},\mathcal{D})}$ in the malicious model.*

7 Future Work

Going forward, we plan to apply our privacy-preserving bartering protocol in the context of organ donation (e.g., for kidney donor exchanges) in order to address the privacy and transparency challenges that the field is currently facing.

References

1. BarterQuest. `http://www.barterquest.com`.
2. R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
3. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty Computation from Threshold Homomorphic Encryption. Technical report, 2000.
4. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty Computation from Threshold Homomorphic Encryption. In *Advances in Cryptology - EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques*, pages 280 – 300. Springer Berlin Heidelberg, 2001.
5. M. Dahl, C. Ning, and T. Toft. On Secure Two-Party Integer Division. In *Financial Cryptography and Data Security: 16th International Conference, FC 2012*, pages 164–178. Springer Berlin Heidelberg, 2012.
6. F. Förg, D. Mayer, S. Wetzel, S. Wüller, and U. Meyer. A Secure Two-Party Bartering Protocol Using Privacy-Preserving Interval Operations. In *Twelfth Annual International Conference on Privacy, Security and Trust*, pages 57–66, 2014.
7. P.-A. Fouque, G. Poupard, and J. Stern. Sharing Decryption in the Context of Voting or Lotteries. In *Financial Cryptography: 4th International Conference, FC 2000*, pages 90–104. Springer Berlin Heidelberg, 2001.
8. J. Garay, B. Schoenmakers, and J. Villegas. Practical and Secure Solutions for Integer Comparison. In *Public Key Cryptography - PKC 2007: 10th International Conference on Practice and Theory in Public-Key Cryptography*, pages 330–342. Springer Berlin Heidelberg, 2007.
9. J. Groth. A Verifiable Secret Shuffe of Homomorphic Encryptions. In *Public Key Cryptography - PKC 2003: 6th International Workshop on Practice and Theory in Public Key Cryptography*, pages 145–160. Springer Berlin Heidelberg, 2002.
10. J. Guajardo, B. Mennink, and B. Schoenmakers. Modulo Reduction for Paillier Encryptions and Application to Secure Statistical Analysis. In *Financial Cryptography and Data Security: 14th International Conference, FC 2010*, pages 375–382. Springer Berlin Heidelberg, 2010.
11. V. Katasonov. Moneyless Business: Barter Transactions Increase Worldwide, 2016. `http://www.strategic-culture.org/news/2016/02/21/moneyless-business-barter-transactions-increase-worldwide.html`.
12. N. López, M. Núñez, I. Rodrguez, and F. Rubio. A Multi-agent System for e-Barter Including Transaction and Shipping Costs. In *Proceedings of the 2003 ACM Symposium on Applied Computing*, pages 587–594. ACM, 2003.
13. A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.
14. P. Paillier. Public-key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology - EUROCRYPT '99: International Conference on the Theory and Application of Cryptographic Techniques*, pages 223–238. Springer Berlin Heidelberg, 1999.

15. B. Schoenmakers and P. Tuyls. Efficient Binary Conversion for Paillier Encrypted Values. In *Advances in Cryptology - EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 522–537. Springer Berlin Heidelberg, 2006.
16. A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.
17. TradeYa. `http://www.tradeya.com/`.
18. U-Exchange. `http://www.u-exchange.com`.
19. S. Wüller, D. Mayer, F. Förg, S. Schüppen, B. Assadsolimani, U. Meyer, and S. Wetzel. Designing Privacy-Preserving Interval Operations Based on Homomorphic Encryption and Secret Sharing Techniques. *Journal of Computer Security*, 25(1):59–81, 2017.
20. S. Wüller, U. Meyer, F. Förg, and S. Wetzel. Privacy-Preserving Conditional Random Selection (Extended Version). In *Thirteenth Annual Conference on Privacy, Security and Trust*, pages 44–53, 2015.
21. S. Wüller, U. Meyer, and S. Wetzel. Towards Privacy-Preserving Multi-Party Bartering. Technical Report AIB-2016-10, RWTH Aachen, 2016.
22. S. Wüller, U. Meyer, and S. Wetzel. Towards Privacy-Preserving Multi-Party Bartering. In *Financial Cryptography and Data Security: FC 2017 International Workshops*. Springer Berlin Heidelberg, 2017. To appear.
23. S. Wüller, W. Pessin, U. Meyer, and S. Wetzel. Privacy-Preserving Two-Party Bartering Secure Against Active Adversaries. In *Fourteenth Annual Conference on Privacy, Security and Trust*, pages 229–238, 2016.

Appendix

In the following, we build on the bartering example from Section 4.1 in order to illustrate the functioning of our bartering protocol. The example considers four parties P_v ($v \in \{1,\dots,4\} = \mathscr{P}$) with private inputs $\mathbf{q}^{(v)}$, where $\mathtt{A},\mathtt{B},\mathtt{C} \in \mathscr{C}$ and $\mathbf{q}^{(1)} = ((\mathtt{A},10),(\mathtt{C},5))$, $\mathbf{q}^{(2)} = ((\mathtt{B},4),(\mathtt{A},6))$, $\mathbf{q}^{(3)} = ((\mathtt{C},10),(\mathtt{B},3))$, and $\mathbf{q}^{(4)} = ((\mathtt{C},8),(\mathtt{A},4))$. The corresponding compatibility graph G^{C} is depicted in Figure 1. Let $\mathcal{W}(G^{\mathrm{TPC}} = (V,E)) = |E|$ and let $\mathcal{D}$ be the uniform distribution. The example considers a public set of trade partner constellation graphs with only four elements $\mathbb{G}^{\mathrm{TPC}} := \{G_1^{\mathrm{TPC}}, G_2^{\mathrm{TPC}}, G_3^{\mathrm{TPC}}, G_4^{\mathrm{TPC}}\}$, where $V = \{1,2,3,4\}$ and

$$
\begin{aligned}
G_1^{\mathrm{TPC}} &:= (V, \{(1,2),(2,3),(3,4),(4,1)\}),\\
G_2^{\mathrm{TPC}} &:= (V, \{(1,2),(2,4),(4,1)\}),\\
G_3^{\mathrm{TPC}} &:= (V, \{(1,2),(2,3),(3,1)\}),\\
G_4^{\mathrm{TPC}} &:= (V, \{(1,4),(4,1)\}).
\end{aligned}
$$

For the given private inputs and the public set of trade partner constellation graphs, the goal of our privacy-preserving bartering protocol is to obliviously compute an actual trade graph which maximizes the welfare function.

In the input phase of protocol $\pi_{\mathrm{MB}}^{(\mathcal{W},\mathcal{D})}$, each party distributes its encrypted private input and proves plaintext knowledge as well as input consistency (see

Section 6). During the construction phase, all parties jointly establish the encrypted adjacency matrix $[\![A]\!]$ of G^{C}:

$$[\![A]\!] = \begin{pmatrix} - & [\![1]\!] & [\![0]\!] & [\![1]\!] \\ [\![0]\!] & - & [\![1]\!] & [\![0]\!] \\ [\![1]\!] & [\![0]\!] & - & [\![0]\!] \\ [\![1]\!] & [\![0]\!] & [\![0]\!] & - \end{pmatrix}$$

In the negotiation phase, the parties jointly compute a second encrypted matrix $[\![A']\!]$. For each pair $(P_v, P_{v'})$ $(v, v' \in \mathscr{P}, v \neq v')$, $[\![A']\!]$ contains the quantity $q_{c_o^{(v)}}^{(v,v')}$ of commodity $c_o^{(v)}$ which P_v has to send to $P_{v'}$ in case that (v, v') corresponds to an edge in the actual trade graph G^{AT} determined by the protocol. In case that there exists an edge (v, v') in G^{C}, i.e., $a_{v,v'} = 1$, $[\![a'_{v,v'}]\!] := [\![q_{c_o^{(v)}}^{(v,v')}]\!]$ is computed by calling gate $\rho^{\omega_\Delta}_{\text{RSI-}\mathcal{D}}$ on inputs $[\![\underline{q}_d'^{(v')}]\!]$ and $[\![\overline{q}_o'^{(v)}]\!]$ with $\underline{q}_d'^{(v')} = \underline{q}_d^{(v')}$ and $\overline{q}_o'^{(v)} = \overline{q}_o^{(v)}$. Otherwise, i.e., $a_{v,v'} = 0$, $\rho^{\omega_\Delta}_{\text{RSI-}\mathcal{D}}$ is called on inputs $[\![\underline{q}_d'^{(v')}]\!]$ and $[\![\overline{q}_o'^{(v)}]\!]$ with $\underline{q}_d'^{(v')} = \overline{q}_o'^{(v)} = 0$. Note that on inputs $[\![0]\!], [\![0]\!]$, gate $\rho^{\omega_\Delta}_{\text{RSI-}\mathcal{D}}$ returns a fresh encryption of 0. For our example let

$$[\![A']\!] = \begin{pmatrix} - & [\![6]\!] & [\![0]\!] & [\![7]\!] \\ [\![0]\!] & - & [\![4]\!] & [\![0]\!] \\ [\![8]\!] & [\![0]\!] & - & [\![0]\!] \\ [\![8]\!] & [\![0]\!] & [\![0]\!] & - \end{pmatrix}$$

where $a'_{1,2} \leftarrow_{\$} [6, 10], a'_{1,4} \leftarrow_{\$} [4, 10], a'_{2,3} \leftarrow_{\$} [3, 4], a'_{3,1} \leftarrow_{\$} [5, 10], a'_{4,1} \leftarrow_{\$} [5, 8]$ assuming that $\mathcal{D}$ corresponds to the uniform distribution.

In the evaluation phase, the parties obliviously check whether or not $G_j^{\mathrm{TPC}} \sqsubseteq G^{\mathrm{C}}$ for each trade partner constellation graph $G_j^{\mathrm{TPC}} \in \mathbb{G}^{\mathrm{TPC}}$ by jointly multiplying the entries of $[\![A]\!]$ which correspond to an edge in G_j^{TPC}. Since only G_3^{TPC} and G_4^{TPC} are subgraphs of G^{C}, the encrypted vector $[\![L]\!]$ is determined as

$$[\![L]\!] := ([\![e_1]\!], [\![e_2]\!], [\![e_3]\!], [\![e_4]\!]) = ([\![0]\!], [\![0]\!], [\![1]\!], [\![1]\!]).$$

Subsequently, each party locally multiplies each entry in $[\![L]\!]$ with the welfare of the corresponding trade partner constellation graph using the homomorphic scalar multiplication operation. For our example, we assume that the welfare of a trade partner constellation graph equals the number of corresponding edges as motivated in Section 4.1. At the end of the prioritization phase, each party holds the encrypted vector

$$[\![L_0]\!] := ([\![0]\!] \times_h 4, [\![0]\!] \times_h 3, [\![1]\!] \times_h 3, [\![1]\!] \times_h 2) = ([\![0]\!], [\![0]\!], [\![3]\!], [\![2]\!]).$$

During the mapping phase, all parties jointly generate four encrypted vectors $[\![L_v]\!]$ (one vector for each party) where the j-th entry of vector $[\![L_v]\!]$ contains the trade partners $N_{o,d}^{(v)}(G_j^{\mathrm{TPC}})$ of P_v as well as the quantities to be sent and

received for the underlying trade: $(q^{(N_o^{(v)},v)}_{c_o^{(N_o^{(v)})}}, q^{(v,N_d^{(v)})}_{c_o^{(v)}})$. Note that for the case that $N_{o,d}^{(v)}(G_j^{\mathrm{TPC}}) = (0,0)$, it holds that both of these quantities are equal to zero.

$$\begin{aligned}
[\![L_1]\!] &:= ([\![(4,2),8,6]\!], [\![(4,2),8,6]\!], [\![(3,2),8,6]\!], [\![(4,4),8,7]\!])\\
[\![L_2]\!] &:= ([\![(1,3),6,4]\!], [\![(1,4),6,0]\!], [\![(1,3),6,4]\!], [\![(0,0),0,0]\!])\\
[\![L_3]\!] &:= ([\![(2,4),4,0]\!], [\![(0,0),0,0]\!], [\![(2,1),4,8]\!], [\![(0,0),0,0]\!])\\
[\![L_4]\!] &:= ([\![(3,1),0,8]\!], [\![(2,1),0,8]\!], [\![(0,0),0,0]\!], [\![(1,1),7,8]\!])
\end{aligned}$$

The main step of the selection phase is the call of gate $\rho_{CRS}^{i^*=0}$ on the joint input $([\![L_0]\!], [\![L_1]\!], \ldots, [\![L_4]\!])$. Considering the table consisting of row vectors $[\![L_0]\!], \ldots, [\![L_4]\!]$

	G_1^{TPC}	G_2^{TPC}	G_3^{TPC}	G_4^{TPC}
$[\![L_0]\!]$	$[\![l_{0,1}]\!]$	$[\![l_{0,2}]\!]$	$[\![l_{0,3}]\!]$	$[\![l_{0,4}]\!]$
$[\![L_1]\!]$	$[\![l_{1,1}]\!]$	$[\![l_{1,2}]\!]$	$[\![l_{1,3}]\!]$	$[\![l_{1,4}]\!]$
$[\![L_2]\!]$	$[\![l_{2,1}]\!]$	$[\![l_{2,2}]\!]$	$[\![l_{2,3}]\!]$	$[\![l_{2,4}]\!]$
$[\![L_3]\!]$	$[\![l_{3,1}]\!]$	$[\![l_{3,2}]\!]$	$[\![l_{3,3}]\!]$	$[\![l_{3,4}]\!]$
$[\![L_4]\!]$	$[\![l_{4,1}]\!]$	$[\![l_{4,2}]\!]$	$[\![l_{4,3}]\!]$	$[\![l_{4,4}]\!]$

where the j-th column is associated with G_j^{TPC}, gate $\rho_{CRS}^{i^*=0}$ returns the column with $max_{j \in \mathbb{N}_{|\mathbb{G}^{\mathrm{TPC}}|}}(l_{0,j})$ (for the case that the maximum is not unique, the output is chosen uniformly at random from those columns with a maximum indicator entry). For our example we have that gate $\rho_{CRS}^{i^*=0}$ returns $([\![l_0^*]\!], [\![l_1^*]\!], \ldots, [\![l_4^*]\!]) := ([\![3]\!], [\![(3,2),8,6]\!], [\![(1,3),6,4]\!], [\![(2,1),4,8]\!], [\![(0,0),0,0]\!])$ which corresponds to the third column of the table. Since $l_0^* \neq 0$, in Step 7.3 of Protocol 1 $[\![b]\!]$ corresponds to an encryption of zero, and thus $[\![o_v]\!] := [\![l_v^*]\!]$.

In the output phase, all parties jointly decrypt $([\![o_1]\!], [\![o_2]\!], [\![o_3]\!], [\![o_4]\!])$ such that P_v learns its local view o_v of the actual trade ($v \in \mathscr{P}$). In the example, e.g., party P_1 learns that it is to receive 8 units of its demanded commodity `C` from P_3 and that it has to send 6 units of its offered commodity `A` to P_2.