RWTH Aachen

Department of Computer Science
Technical Report

Towards Privacy-Preserving Multi-Party Bartering

Stefan Wüller, Ulrike Meyer and Susanne Wetzel

ISSN 0935–3232 · Aachener Informatik-Berichte · AIB-2016-10

RWTH Aachen · Department of Computer Science · December 2016

The publications of the Department of Computer Science of RWTH Aachen University are in general accessible through the World Wide Web.

http://aib.informatik.rwth-aachen.de/

Towards Privacy-Preserving Multi-Party Bartering

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

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

Abstract. Both B2B bartering as well as bartering between individuals is increasingly facilitated through online platforms. However, typically these platforms lack automation and tend to neglect the privacy of their users by leaking crucial information about trades. It is in this context that we devise the first privacy-preserving protocol for automatically determining an actual trade between multiple parties without involving a trusted third party.

1 Introduction

The Encyclopedia Britannica defines *bartering* as "the direct exchange of goods or services—without an intervening medium of exchange or money—either according to established rates of exchange or by bargaining" [Bri]. According to [min], the first mentioning of bartering dates back thousands of years. Throughout history, the relevance of bartering increased in times of crises. For example, in the US bartering was of utmost importance during the great depression in the 1930s, as well as during a long recession in the early 1980s [Sto]. Coherently, in Europe bartering economies are currently reported to flourish in countries like Greece due to imminent liquidity problems [Ald15]. However, the popularity of bartering is not only increasing as a result of economic crises. Instead individuals as well as businesses increasingly engage in bartering in recent years. In fact, both the National Association for Trade Exchanges [oTE] as well as the International Reciprocal Trade Association [IRT] report that well over 400,000 companies worldwide currently engage in bartering. This high interest in bartering is also due to the fact that bartering platforms such as *U-Exchange*, *BarterQuest*, or *TradeYa* [UE,Bar,Tra,Qui] increasingly facilitate bartering between individuals as well as businesses. These platforms typically provide functionalities such as allowing users to specify offers and demands, searching, browsing, and filtering the offers and demands of other users, or supporting bargaining on the final exchange rates, e.g., by offering private chat rooms for the parties involved in a trade (cf. [Boc11]). However, most of these platforms lack automation, i.e., users have to actively search other users' offers or demands, or have to initiate contact with others in order to bargain the final exchange rates. Even more importantly, these platforms do typically not very well protect the privacy of users. In particular, they typically disclose what users seek or offer to other users even if a trade between these users is not possible. Moreover, these platforms often force users to disclose there limits on the exchange rates they are willing to accept such that their room to negotiate is unnecessarily restricted.

For the two-party case, these privacy issues are addressed with the help of secure multi-party computation (SMPC), e.g., in [FO08,FMW+14]. While these two party protocols can obviously be used to find pairwise trades in the multi-party setting with more than two parties as well, they cannot be used to determine trade cycles between more than two parties. In this case, each party should only learn which quantity of which commodity it is to give to which other party and which quantity of which commodity it receives from which other party in return. Any other information on what is traded at which exchange rates between the other parties in the trade cycle is to remain private. The particular challenge of finding such cycles in a privacy-preserving way in the SMPC setting has already been recognized in [FT98] but has to the best of our knowledge not been addressed so far. In [KMRR15], multi-party bartering is addressed under a different notion of privacy, namely marginal differential privacy. Compared to the SMPC setting, this setting is weaker in that it requires a trusted third party in order to determine an actual trade and assumes non-colluding corrupted parties.

As a first step towards privacy-preserving multi-party bartering, we design a secure multi-party bartering process in which each party's demand is fulfilled by at most one other party and each party is allowed to specify exactly one quote defining its offered and desired commodities and quantities. For a given set of parties and their quotes our bartering process automatically and securely determines an actual trade comprising the actual trade constellation of the parties (i.e., which party trades with which other party) as well as the actual commodities and quantities to be traded. The actual trade can be selected based on different selection strategies including the maximization of the number of parties able to trade. Throughout the bartering process, each party keeps its quote secret at all times. Upon completion of the bartering process, each party learns nothing but its local view of the actual trade which includes its trade partners as well as the commodities and quantities to be sent and received. At the core of the newly designed bartering process is a protocol that securely determines the actual trade constellation. This protocol makes use of two main ideas. First, it generalizes the idea of intertwining comparison and scalar product computations used in [FMW+14]. Second, it uses a novel privacy-preserving mapping operation that is based on the uniqueness of prime factorization, which is of independent interest.

The remainder of this paper is organized as follows. After reviewing related work in Section 2, we introduce definitions and notations used throughout the paper (Section 3). We then devise a multi-party bartering terminology and provide an intuition for the operation of our protocols (Section 4). In Section 5, we review existing protocols used as building blocks in the context of our work and present a novel protocol for obliviously determining whether or not a specific trade constellation corresponds to a potential trade. Building on that, we introduce our novel protocol for determining an actual trade constellation (Section 6). The paper closes with some remarks on future work.

2 Related Work

Our work extends on the privacy-preserving bartering protocol for two parties by Foerg et al. [FMW+14]. Their protocol allows two parties to specify an offered and a desired commodity along with the corresponding quantities and determines whether or not (and if so at what rate) the two parties are willing to mutually exchange their commodities. Our work presented in this paper extends their setting to the multi-party case. Since the computation of an actual trade for an arbitrary set of parties involves the identification of complex exchange structures like trade cycles, it is necessary to extend the available terminology and to devise novel techniques and protocols along with the corresponding security proofs.

To the best of our knowledge, there is only one approach to privacy-preserving multi-party bartering that has been proposed in the past [KMRR15]: Kannan et al. introduce a protocol where each party holds a (indivisible) commodity from a publicly known finite set of commodities as well as a totally ordered preference list over all commodities in the set. Their goal is then to determine an actual trade between multiple parties such that the computed commodity allocation is pareto optimal while the input of each party (commodity and preference list) is kept private. Specifically, the protocol protects the parties' input under the notion of *marginal differential privacy* [KMRR15] which is a relaxation of *differential privacy* [Dwo06]. In contrast to differential privacy, marginal differential privacy is restricted to an adversary that has access to the protocol output of only one single party which corresponds to the assumption that there are no colluding parties participating in the protocol which try to subvert the privacy of another party. The approach by Kannan et al. differs from our approach in that it requires a trusted third party in order to determine an actual trade and in that they use a weaker privacy notion that assumes non-colluding parties. Furthermore, their approach supports only indivisible commodities. In our approach, an actual trade is computed without the help of a trusted third party and we allow that all but one colluding parties may be controlled by an adversary. In addition, our approach supports divisible commodities.

In contrast to e-commerce (and auctions), bartering transactions are not necessarily reduced to money which allows for a richer structure of exchanges [LNRR03]: A trade takes place if the involved parties are satisfied w.r.t. the specification of their offered and desired commodities and the corresponding quantities. If the commodities first have to be converted into money (as it is the case for e-commerce and auctions), the prices of the commodities have to be individually determined. Consequently, a party desiring a commodity which is more expensive than its offered commodity is not able to barter, although a trade could have taken place if the commodities were traded directly [LNRR03]. Thus, privacy-preserving protocols for e-commerce scenarios (e.g., [ABO04,EA05]) or auctions (e.g., [Bra02,NSY04]) are not suitable in the context of privacy-preserving bartering.

In order to design our privacy-preserving bartering process, we make use of SMPC techniques. While there exist generic SMPC frameworks providing general results in the sense that they allow for any function to be securely computed [Yao82,BOGW88,GMW87], using these frameworks in a straight-forward fashion to design a specific functionality can lead to inefficient solutions [FO08], [BCD+09]. By contrast, the specific purpose design of complex SMPC protocols

allows to provide domain specific solutions and to involve additional building blocks like intermediate decryption which in turn allows to design more efficient protocols [BCD+09,CT12,AMP04,HL10]. Therefore, we focus on designing special purpose SMPC protocols rather than using any generic solutions.

3 Preliminaries

In the following, we introduce notations and recall definitions used throughout the paper.

3.1 Notation

By $s \leftarrow_{\$} S$ we indicate that s is drawn uniformly at random from S. $\mathbb{N}_u := \{1, ..., u\}$ refers to the set of natural numbers less than or equal to $u \in \mathbb{N}$. The set of all prime numbers within an integer interval I is referred to as $\mathbf{P}_I$. We denote the index set of all parties P_i participating in a multi-party protocol as $\mathscr{P} := \{1, \ldots, \iota\}$ where $i \in \mathscr{P}$. Furthermore, λ denotes the empty string.

3.2 The Paillier Threshold Cryptosystem

Our privacy-preserving protocols require an additively homomorphic cryptosystem which is semantically secure [KL07] against chosen-plaintext attacks and provides a (τ, ι) threshold variant, i.e., the decryption key is distributed amongst ι parties such that at least $\tau \leq \iota$ parties have to collaborate in order to decrypt a ciphertext.

In the following, we summarize the (τ, ι) threshold variant of the Paillier cryptosystem [Pai99] from [DJ01] along with the Paillier-related notation used throughout the paper.[3]

The public key corresponds to an RSA modulus $N = p \cdot q$ of bit length k, where p, q are safe primes (i.e., there are prime numbers p' and q' such that $p = 2p' + 1$ and $q = 2q' + 1$) and k refers to the security parameter. The private key $d \in \mathbb{Z}_{p'q'N^s}$ with $s > 0$, $s \in \mathbb{N}$ satisfying $d = 0 \ mod \ p'q'$ and $d = 1 \ mod \ N^s$ is polynomially shared between $P_1, \ldots, P_\iota$ such that at least τ parties have to cooperate for decryption.[4] The encryption of a message m in the *plaintext space* $\mathbb{P} := \mathbb{Z}_{N^s}$ is computed as $c = E(m) := (N+1)^m r^{N^s} \ mod \ N^{s+1}$ where $r \leftarrow_{\$} \mathbb{Z}^*_{N^{s+1}}$ and c is an element in the *ciphertext space* $\mathbb{C} := \mathbb{Z}^*_{N^{s+1}}$. Throughout the paper we assume $s = 1$. We have that the plaintext space $\mathbb{P}$ forms the additive group $(\mathbb{Z}_N, +)$, and the ciphertext space $\mathbb{C}$ forms the multiplicative group $(\mathbb{Z}^*_{N^2}, \cdot)$. For further details we refer to [DJ01].

Let $m, m_1, m_2 \in \mathbb{P}$ and $\kappa \in \mathbb{N} \setminus \{0\}$. The Paillier ($(\tau, \iota)$ threshold) cryptosystem provides for *homomorphic addition*

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

and *homomorphic scalar multiplication*

$$E(m) \times_h \kappa := \underbrace{E(m) \cdot E(m) \cdots E(m)}_{\kappa \text{ times}} = E(\kappa \cdot m).$$

[3] Other threshold variants, e.g., the one from [FPS01] can be used as well.

[4] Note that there exist multi-party computation techniques allowing to generate and distribute Paillier keys without the help of a trusted third party [DJ01,NS11].

Furthermore, we write $E(m_1) -_h E(m_2)$ for

$$E(m_1) +_h (E(m_2))^{-1} = E(m_1 - m_2)$$

where $E(m_2)^{-1}$ denotes the multiplicative inverse of $E(m_2)$ in $\mathbb{C}$. In addition, we write $E(m) := c$ for $m := D(c)$.

For all $E(m_1)$, $E(m_2) \in \mathbb{C}$, the homomorphic multiplication operation $\cdot_h$ is defined as

$$E(m_1) \cdot_h E(m_2) := E(m_1 \cdot m_2).$$

The computation of $E(m_1 \cdot m_2)$ can be carried out using a homomorphic scalar multiplication in case m_1 or m_2 is known. A ciphertext $E(m)$ can be *randomized* (or *re-randomized*) by computing $E(m) +_h E(0)$. For a vector $V = (v_1, ..., v_n)$ we write $E(V)$ to denote the encryption of each entry v_i ($i \in \mathbb{N}_n$). Similarly, we write $Rnd(E(V))$ to denote the randomization of each encrypted vector entry, where for each randomization a fresh encryption of 0 is used.

For the remaining sections $\mathbb{P}$, $\mathbb{C}$, and $E(\cdot)$ refer to the plaintext space, the ciphertext space, and the encryption function of (τ, ι) threshold Paillier, respectively.

3.3 Definition of Security

In order to define security comprising privacy and correctness, we have to specify the capabilities of an adversary under whose presence a protocol has to be secure. We prove our protocols to be secure in the semi-honest model. A semi-honest adversary controls a set of corrupted parties which correctly follow the protocol specification with the exception that each corrupted party keeps record of all data it generates itself and all messages it receives from other parties. The postulation of the semi-honest model for application as well as for research is well justified [Kol06,Gol09]. Generally, for real-world applications it suffices to provide security against semi-honest adversaries especially when the essential protocol behavior is controlled by a complex software [Kol06]. Furthermore, designing protocols for the semi-honest model is a first step towards designing protocols for the malicious model where the adversary can arbitrarily deviate from the protocol specification.[5]

We assume that the parties communicate over authentic channels, i.e., the transferred data is resistant to tampering but can be wiretapped.[6]

Let $\widehat{X} := (X_1, \ldots, X_\iota)$ and let $\mathcal{F} : (\{0,1\}^*)^\iota \to (\{0,1\}^*)^\iota$, $\widehat{X} \mapsto (\mathcal{F}_1(\widehat{X}), ..., \mathcal{F}_\iota(\widehat{X}))$ be a multi-party ($|\mathscr{P}| = \iota \geq 2$) functionality computable in polynomial time where P_i provides input X_i and obtains output $\mathcal{F}_i(\widehat{X})$ ($i \in \mathscr{P}$). Let π be an ι-party protocol for computing functionality $\mathcal{F}$. We write $I_C := \{i_1, ..., i_\kappa\} \subset \mathscr{P}$ for the index set of $1 \leq \kappa < \iota$ corrupted parties controlled by the adversary. The view of P_i during an execution of π on input $\widehat{X}$ and security parameter s is denoted as $\text{VIEW}_i^\pi(s, \widehat{X}) := (s, X_i, \mathring{r}_i, m_{i,1}, ..., m_{i,n})$, where

[5] In case of using threshold homomorphic encryption as underlying SMPC technique, starting from a protocol providing security in the semi-honest model, security in the malicious model can be obtained without increasing the protocol complexity [CDN01]. This generally does not hold for other SMPC techniques like secret sharing.

[6] Note that there exist standard techniques allowing that the parties do not have to know each other to jointly execute an SMPC protocol, e.g., by differentiating between input parties and computing parties.

$\mathring{r}_i$ represents P_i's internal random tape and $m_{i,j}$ represents the j-th message P_i received during a protocol execution of π. We write $\text{OUTPUT}^\pi(s, \widehat{X}) := (\text{OUTPUT}_1^\pi(s, \widehat{X}), \ldots, \text{OUTPUT}_\iota^\pi(s, \widehat{X}))$ in order to refer to the output of protocol π on input $\widehat{X}$ and security parameter s. Let $\widehat{X}_{I_C}$, $\mathcal{F}_{I_C}(\widehat{X})$, and $\text{VIEW}_{I_C}^\pi(\widehat{X})$ denote the κ-tuples $(X_{i_1}, ..., X_{i_\kappa})$, $(\mathcal{F}_{i_1}(\widehat{X}), ..., \mathcal{F}_{i_\kappa}(\widehat{X}))$, and $(I_C, \text{VIEW}_{i_1}^\pi(\widehat{X}), ..., \text{VIEW}_{i_\kappa}^\pi(\widehat{X}))$, respectively.

Definition 1. (Security: Semi-Honest Model, Multi-Party Setting [Gol09]). *π securely computes functionality $\mathcal{F}$ if there exists a probabilistic polynomial time algorithm $\mathcal{S}$ such that for every I_C it holds that $\{(\mathcal{S}(1^s, I_C, \widehat{X}_{I_C}, \mathcal{F}_{I_C}(\widehat{X})), \mathcal{F}(\widehat{X}))\}_{\widehat{X},s}$ and $\{(VIEW_{I_C}^\pi(\widehat{X}, s), OUTPUT^\pi(s, \widehat{X}))\}_{\widehat{X},s}$ are computational indistinguishable.*

In the following, we assume that the security parameter s is implicitly given and sufficiently large and for matters of convenience omit it from the remaining considerations. We call $\mathcal{S}$ a *simulator* and enclose the values it *simulates* (on input $\widehat{X}_{I_C}$ and $\mathcal{F}_{I_C}(\widehat{X})$ to generate a view which is computationally indistinguishable from $\text{VIEW}_{I_C}^\pi(\widehat{X})$) by square brackets $\langle\cdot\rangle$ in order to distinguish between simulated values and those occurring during a protocol run. If not stated otherwise, $\mathcal{S}$ sets $\langle X_c \rangle := X_c$ $(c \in I_C)$.

In order to facilitate the security proof of a protocol π implementing functionality $\mathcal{F}$ where π consists of a finite set of sub-protocols $\rho_1, \ldots, \rho_n$ securely computing functionalities $\mathcal{G}_1, \ldots, \mathcal{G}_n$ in the semi-honest model, we can apply the *Modular Composition Theorem* [Can00] which states that if π' securely computes $\mathcal{F}$ in the semi-honest model where the sub-protocol calls of π are replaced by calls to a trusted third party computing $\mathcal{G}_1, \ldots, \mathcal{G}_n$, then π securely computes $\mathcal{F}$ in the semi-honest model.

To prove our protocols to be secure in the semi-honest model, we proceed in two steps as it is done in [AL11]. First, we prove that $\{\mathcal{F}(\widehat{X})\}_{\widehat{X}} \overset{c}{\equiv} \{OUTPUT^\pi(\widehat{X})\}_{\widehat{X}}$. We refer to this step as *Correct Output Distribution* (COD). Second, we prove that $\text{VIEW}_{I_C}^\pi$ can be simulated under consideration of the given inputs and outputs of all corrupted parties such that $\text{VIEW}_{I_C}^\pi$ and the corresponding simulated view are computationally indistinguishable, referred to as *Correct View Distribution* (CVD). If the output of $\mathcal{F}$ is encrypted with a probabilistic cryptosystem, such as Paillier, we also have to prove that the distribution of the decrypted output values of π are computationally indistinguishable from the distribution of the decrypted output values prescribed by the definition of functionality $\mathcal{F}$. We refer to this step as *Correct Distribution of Decrypted Output* (CDDO). Note, that this step is necessary to assure the correctness of π and that it is not captured by Definition 1.

To refer to a concrete functionality or protocol, we use the templates $\mathcal{F}_{name}^{[affix]}$ and $\pi_{name}^{[affix]}$ where protocol $\pi_{name}^{[affix]}$ is an implementation of functionality $\mathcal{F}_{name}^{[affix]}$ with *name* and *affix* describing the functionality to be computed where the use of *affix* is optional. For convenience, we omit *name* and *affix* for the case that the target functionality and protocol is clear from the context. Furthermore, we write $\mathcal{F}(X_1, \ldots, X_\iota, X)$ to denote that X is a public input that is known by all parties. $(o) \leftarrow \mathcal{F}(X)$ indicates that all parties have common input X and common output o.

$TPT(S)$	Trade Partner Tuple (Set)	Def. 2 (below Def. 12)
$TPC(S)$	Trade Partner Constellation (Set)	Def. 3 (below Def. 4)
$PTPC(S)$	Potential Trade Partner Constellation (Set)	Def. 4 (below Def. 4)
$ATPC$	Actual Trade Partner Constellation	Def. 5
AT	Actual Trade	Def. 6

Table 1: Bartering related acronyms used throughout the paper.

4 Overview

4.1 Bartering Related 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. In this paper, we focus on so-called $(1:1)$ *trades* with one offered and one desired commodity for each party. In such a trade, each party receives some quantity of its desired commodity from at most one party and sends some quantity of its offered commodity to at most one other party.

More specifically, we consider a set of ι parties $\{P_i | i \in \mathscr{P}\}$ with $\mathscr{P} := \mathbb{N}_\iota$ and a publicly known finite set $\mathscr{C} := \{c_1, \ldots, c_n\}$ of divisible commodities. Each party P_i specifies exactly one *quote* $\mathbf{q}^{(i)} := (\mathbf{o}^{(i)}, \mathbf{d}^{(i)})$ where $\mathbf{o}^{(i)}$ and $\mathbf{d}^{(i)}$ is P_i's *offer* and *demand*, respectively. We model $\mathbf{o}^{(i)}$ as a 3-tuple $\mathbf{o}^{(i)} := (c_o^{(i)}, \underline{q}_o^{(i)}, \overline{q}_o^{(i)})$ where $c_o^{(i)} \in \mathscr{C}$ specifies the commodity offered by P_i and $\underline{q}_o^{(i)} \in \mathbb{N}\backslash\{0\}$ ($\overline{q}_o^{(i)} \in \mathbb{N}\backslash\{0\}$) denotes the minimum (maximum) quantity of $c_o^{(i)}$ offered. Similarly, we model $\mathbf{d}^{(i)} := (c_d^{(i)}, \underline{q}_d^{(i)}, \overline{q}_d^{(i)})$ with $c_d^{(i)} \in \mathscr{C}$ and $\underline{q}_d^{(i)}, \overline{q}_d^{(i)} \in \mathbb{N}\backslash\{0\}$. With $\mathbf{q}^{(i)}$ a party P_i indicates that it is *satisfied* with a trade if it receives at least $\underline{q}_d^{(i)}$ and at most $\overline{q}_d^{(i)}$ units of commodity $c_d^{(i)}$ and sending at least $\underline{q}_o^{(i)}$ and at most $\overline{q}_o^{(i)}$ units of $c_o^{(i)}$. For convenience, we assume that $\underline{q}_o^{(i)} = 1$ and $\overline{q}_d^{(i)} = \infty$. The *quantity ranges* of the offered and desired commodities of a party P_i ($i \in \mathscr{P}$) are thus defined as $Q_o^{(i)} := [1, \overline{q}_o^{(i)}]$ and $Q_d^{(i)} := [\underline{q}_d^{(i)}, \infty]$. We write $q_{c_o^{(i)}}^{(i,i')}$ in order to indicate at which quantity $P_{i'}$ will receive commodity $c_o^{(i)}$ from P_i ($i, i' \in \mathscr{P}$).

We introduce the following bartering related terms which are summarized in Table 1 and illustrated in Figure 1:

Definition 2. (Trade Partner Tuple). *A* trade partner tuple $TPT^{(i)} := (x^{(i)}, y^{(i)})$ *for* P_i ($i \in \mathscr{P}$) *with* $x^{(i)}, y^{(i)} \in \mathscr{P}\backslash\{i\}$ *is a* 2*-tuple which specifies the indices of the trading partners* $P_{x^{(i)}}$ *and* $P_{y^{(i)}}$ *of* P_i*:* $P_{x^{(i)}}$ *is the offerer of party* P_i*, i.e.,* P_i *receives some quantity of some commodity from* $P_{x^{(i)}}$*, while* $P_{y^{(i)}}$ *is the demander of* P_i*, i.e.,* P_i *has to send some quantity of some commodity to* $P_{y^{(i)}}$*. If a party* P_i *neither sends nor receives any commodity in a trade, i.e., it does not participate, we write* $TPT^{(i)} = (0, 0)$*.*

Definition 3. (Trade Partner Constellation). *A* trade partner constellation $TPC := (TPT^{(1)}, TPT^{(2)}, ..., TPT^{(\iota)})$ *is an* ι*-tuple which specifies exactly one trade partner tuple for each* P_i ($i \in \mathscr{P}$) *and has the following property: for each trade partner tuple* $TPT^{(i)} = (x^{(i)}, y^{(i)})$ *it either holds that* $x^{(i)} = y^{(i)}$ *or it holds that there exist exactly two distinct entries* $TPT^{(i')}$ *and* $TPT^{(i'')}$ *with* $i \neq i', i''$ *such that* $TPT^{(i')} = (y^{(i)}, y^{(i')})$ *and* $TPT^{(i'')} = (x^{(i'')}, x^{(i)})$*.*

Definition 3 ensures that each party that participates as offerer (demander) in some TPT of a TPC also participates as demander (offerer) either in the same or in exactly one other TPT of the TPC.

For a fixed context of quotes $\mathbf{Q} := \{\mathbf{q}^{(1)}, ..., \mathbf{q}^{(\iota)}\}$ with $\mathbf{q}^{(i)} = ((c_o^{(i)}, \underline{q}_o^{(i)}, \overline{q}_o^{(i)}), (c_d^{(i)}, \underline{q}_d^{(i)}, \overline{q}_d^{(i)}))$, a TPC is transformed into a *trade partner constellation formula*, written $\varphi \overset{\mathbf{Q}}{\sim} TPC$, such that:

$$\varphi := \bigwedge_{\substack{i=1 \\ (x^{(i)}, y^{(i)}) \neq (0,0)}}^{\iota} \mathcal{C}(\mathbf{q}^{(i)}, \mathbf{q}^{(x^{(i)})}) \wedge \mathcal{R}(\mathbf{q}^{(i)}, \mathbf{q}^{(x^{(i)})}) \tag{1}$$

with

$$\mathcal{C}(\mathbf{q}^{(a)}, \mathbf{q}^{(b)}) := \begin{cases} 1 & \text{if } (\mathbf{q}^{(a)}, \mathbf{q}^{(b)}) \in C \\ 0 & \text{otherwise} \end{cases} \text{ and}$$

$$\mathcal{R}(\mathbf{q}^{(a)}, \mathbf{q}^{(b)}) := \begin{cases} 1 & \text{if } (\mathbf{q}^{(a)}, \mathbf{q}^{(b)}) \in R \\ 0 & \text{otherwise} \end{cases},$$

where

$$C := \{(\mathbf{q}^{(a)}, \mathbf{q}^{(b)}) | c_d^{(a)} = c_o^{(b)}\} \text{ and}$$

$$R := \{(\mathbf{q}^{(a)}, \mathbf{q}^{(b)}) | \underline{q}_d^{(a)} < \overline{q}_o^{(b)}\}.$$

Evaluating φ (for a given context of quotes) denoted as $[\![\varphi]\!] \in \{0, 1\}$ allows one to check whether or not there is a trade which all parties P_i (with $TPT^{(i)} \neq (0, 0)$) in the corresponding trade partner constellation are satisfied with. The trade partner constellations for which this holds for a given context of quotes $\mathbf{Q}$ are referred to as *potential trade partner constellations*:

Definition 4. (Potential Trade Partner Constellation). *For a context of quotes* $\mathbf{Q}$*, a trade partner constellation* TPC *is a* potential trade partner constellation *(*$PTPC$*), iff* $\varphi \overset{\mathbf{Q}}{\sim} TPC$ *and* $[\![\varphi]\!] = 1$*.*

We write $TPCS := \{TPC_1, \ldots, TPC_t\}$ for a set of trade partner constellations. Given $TPCS$ and $\mathbf{Q}$, the set of potential trade partner constellations is denoted as $PTPCS$. Furthermore, given $TPCS$ and $\mathbf{Q}$, we define $\Phi := \{\varphi_j | \varphi_j \overset{\mathbf{Q}}{\sim} TPC_j, j \in \mathbb{N}_{|TPCS|}\}$ and $\Phi_{\text{sat}} := \{\varphi_j | \varphi_j \in \Phi, [\![\varphi_j]\!] = 1\} \subseteq \Phi$.

Definition 5. (Actual Trade Partner Constellation). *An* actual trade partner constellation $ATPC$ *is a specific* $PTPC$ *drawn from* $PTPCS$ *based on a specified strategy.*

For matters of convenience, we first assume that $ATPC$ is drawn uniformly at random from $PTPCS$. In Section 6.3, we sketch a modification of our protocol allowing to select an $ATPC$ maximizing the number of traded commodities (without reducing the level of privacy). Other optimization criteria can be integrated analogously.

Definition 6. (Actual Trade). *An* actual trade AT *for an* $ATPC$ *specifies the actual commodities and actual quantities for the commodities traded between the parties involved in* $ATPC$*.*

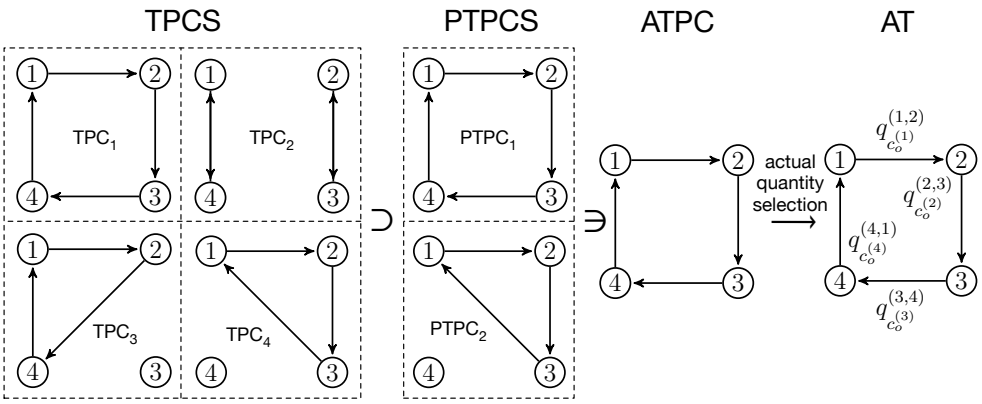

Fig. 1: Illustration of the bartering related terms and their relations.

Figure 1 illustrates the interdependency of the introduced terms. A trade partner constellation can be visualized as a directed graph (see Definition 13, Appendix A), i.e., a node represents a party and a directed edge between two nodes represents the exchange direction of a commodity between two parties. For example, according to the node labels and the direction of the edges we have that TPC_4 in Figure 1 is equal to $(TPT^{(1)}, TPT^{(2)}, TPT^{(3)}, TPT^{(4)}) = ((3,2),(1,3),(2,1),(0,0))$. A potential trade partner constellation set is a subset of a given trade partner constellation set containing those trade partner constellations which form the basis for a trade all involved parties are satisfied with when taking the given context of quotes into account. In Figure 1, we assume a context of quotes such that $TPC_1 = PTPC_1$ and $TPC_4 = PTPC_2$ are potential trade partner constellations. An actual trade partner constellation is an element from the set of potential trade partner constellation selected w.r.t. a specific strategy. In Figure 1, the actual trade partner constellation is chosen such that it maximizes the number of traded commodities (four commodities can be traded in $PTPC_1$ while only three commodities can be traded in $PTPC_2$). The determined actual trade partner constellation is transferred into an actual trade by selecting the actual quantities of the commodities to be traded. In Figure 1, the actual trade indicates that P_1 has to send $q^{(1,2)}_{c_o^{(1)}}$ units of commodity $c_o^{(1)}$ to P_2, that P_2 has to send $q^{(2,3)}_{c_o^{(2)}}$ units of commodity $c_o^{(2)}$ to P_3, and so on.

4.2 Bartering Process and Intuition

The overall goal of a bartering process between parties $P_1, ..., P_\iota$ with a context of quotes $\mathbf{Q} = (\mathbf{q}^{(1)}, ..., \mathbf{q}^{(\iota)})$ is to determine an actual trade, i.e., one specific trade with which all parties are satisfied. Our bartering process introduced in this paper can determine such an actual trade from the set of all possible trade partner constellations. However, for matters of efficiency, it is also possible to use a smaller trade partner constellation set, e.g., one which may contain only trade partner constellations of 5-trade cycles (cf. Appendix A) or constellations in which specific parties get to trade (cf. $TPCS$ in Figure 1). Upon input of the trade partner constellation set, the bartering process tries to find an actual trade consistent with the trade partner constellations in the given trade partner constellation set.

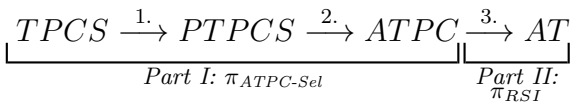

Fig. 2: Illustration of the overall bartering process.

Finding an actual trade first requires the determining of the set of potential trade constellations, i.e., those trade constellations in the trade partner constellation set for which the commodities and quantities of the involved parties in their roles of offerer and demander match (Transition 1, Fig. 2). Subsequently, one of the potential trade constellations is selected as actual trade partner constellation (Transition 1, Fig. 2). This constellation then already indicates which parties will send (and receive) some commodity to (from) which other party in the (yet to be determined) actual trade. Finally, the parties individually engage in a two-party protocol with each one of their trade partners (determined by the actual trade partner constellation) in order to select the actual quantities for the commodities to be traded (Transition 3, Fig. 2).

In order to implement such a bartering process securely, the input of the parties, i.e., their quotes, have to be kept secret throughout the process. Moreover, at the end of the process the parties should learn no more than their local view of the selected actual trade, i.e., their own trade partners and the commodities and quantities to be traded with them. Our newly developed bartering process consists of two parts (cf. Figure 2).

(Part I.) For the first part, we design a multi-party protocol $\pi_{\text{ATPC-Sel}}$ (secure in the semi-honest model) that takes a context of private quotes $\mathbf{Q}$ as well as a (publicly known) set of trade partner constellations as input and then performs the following steps: (1) securely determine the potential trade partner constellation set, (2) securely select an actual trade partner constellation, and (3) provide each party P_i with (nothing but) its actual trade partner tuple in the actual trade partner constellation as output (see Section 6.1).

(Part II.) In the second part, each party is involved in the two-party protocol π_{RSI} with each of its trade partners to determine the actual quantities traded using a two-party protocol for the secure computation of a random sub-interval introduced in [FMW$^+$14] (see Section 6.2).

The first step of the multi-party protocol $\pi_{\text{ATPC-Sel}}$ (determining the potential trade partner constellation set) is implemented with the help of an oblivious evaluation of a trade constellation formula. A protocol solving this task is introduced in Section 5.2 (see Definition 11). It extends on an approach introduced in [FMW$^+$14] in that we generalize the order of securely intertwining comparison and scalar product operations and introduce a secure iteration over these operations thus facilitating the secure matching of quotes for multiple (i.e., more than two) parties.

The second step of the multi-party protocol (selecting an actual trade partner constellation from the potential trade partner constellation set) makes use of the Conditional Random Selection (CRS) protocol introduced in [WMFW15] (see Definition 9, Section 5).

Finally, the third step of the multi-party protocol (which ensures that each party learns nothing but its local view of the selected actual trade partner con-

stellation) makes use of a novel encoding and decoding operation that is based on the uniqueness of prime factorization. Specifically, each party is assigned a unique set of prime numbers. Each party then selects a mapping between its prime numbers and all of its trade partner tuples in any of the trade partner constellations (including those where the respective trade partner tuple is equal to $(0, 0)$) in the trade partner constellation set. Each party keeps its mapping secret. Subsequently, for all trade partner constellations, all parties jointly compute the product of the corresponding prime numbers (locally mapped to the respective trade partner tuples by the parties) in an oblivious fashion thus enabling each party to uniquely encode its participation in the trade partner constellation by contributing a unique prime number in each case. In turn, given the product of prime numbers for the actual trade partner constellation, through trial division each party can locally determine which one of its prime numbers divides the product and then identify the respective trade partner tuple based on its own secret mapping. It is important to note that knowing which set of prime numbers was assigned to which party, anyone can perform the trial division step. However, given the fact that each party keeps its mapping between its prime numbers and its trade partner tuples secret, this does not leak any information. Since also trade partner tuple $(0, 0)$ is mapped to a prime number (individually by each party) and the product of prime numbers includes those prime numbers for parties which neither send nor receive a commodity, the prime number product does not leak which parties participate in the selected actual trade partner constellation. It is also important to note that the size of the prime numbers does not have any influence on the security of our new protocol.

In Appendix A, we provide an example to further illustrate the workings of our new bartering process.

5 Building Blocks

In this section, we first present existing functionalities and the corresponding protocols which are used as building blocks in our novel protocols. Furthermore, we introduce a novel functionality for obliviously evaluating trade partner constellation formulas and present a corresponding protocol which is proven secure in the semi-honest model.

5.1 Existing Building Blocks

Definition 7. ($\mathcal{F}^{\text{LT}}_{\text{SC-SO}}$: Two-party Secure Comparison (SC) Less Than (LT) with Shared Output (SO)). *Let P_1 and P_2 hold integers v_1 and v_2, respectively. Then, functionality $\mathcal{F}^{LT}_{SC\text{-}SO}$ is given by $(w_1, w_2) \leftarrow \mathcal{F}^{LT}_{SC\text{-}SO}(v_1, v_2)$ with $w_2 \leftarrow_{\$} \{0, 1\}$ and $w_1 \in \{0, 1\}$ s.t. $w_1 \oplus w_2$ indicates whether or not $v_1 < v_2$.*

Definition 8. ($\mathcal{F}^{\omega}_{RSI}$: Two-party secure computation of a Random (R) Sub-Interval (SI)). *Let P_1 hold integer interval I_1 and P_2 hold integer interval I_2 such that $\omega \leq |I_1 \cap I_2|$. Then, functionality $\mathcal{F}^{\omega}_{RSI}$ is given by $([l_r, u_r]) \leftarrow \mathcal{F}^{\omega}_{RSI}(I_1, I_2)$ where $[l_r, u_r]$ is a sub-interval drawn uniformly at random from $I_o = I_1 \cap I_2$ s.t. $|[l_r, u_r]| = \omega$.*

Definition 9. ($\mathcal{F}_{\text{CRS-C}}^{i^*}$: Multi-party Conditional (C) Random (R) Selection (S) with output Check (C)). *Let $P_1, \ldots, P_\iota$ hold m vectors $E(L_i) = (E(l_{i,1}), \ldots, E(l_{i,n}))$ of length n of integers $l_{i,j} \in \mathbb{P}$ ($i \in \mathbb{N}_m$, $j \in \mathbb{N}_n$) encrypted with (ι, ι) threshold Paillier. Let $E(L_{i^*})$ be an encrypted binary indicator vector and $\{E(L_1), ..., E(L_m)\} \setminus \{E(L_{i^*})\}$ be the value vectors with $i^* \in \mathbb{N}_m$. Then, functionality $\mathcal{F}_{CRS\text{-}C}^{i^*}$ is given by $((E(o_1), \ldots, E(o_m))) \leftarrow \mathcal{F}_{CRS\text{-}C}^{i^*}((E(L_1), \ldots, E(L_m)))$ with $E(o_i) = Rnd(E(l_{i,j^*}))$ ($i \in \mathbb{N}_m$) where $j^* \leftarrow_\$ \{j \in \mathbb{N}_n : l_{i^*,j} = 1\}$ if there exists at least one $j \in \mathbb{N}_n$ s.t. $l_{i^*,j} > 0$. Otherwise, $\mathcal{F}_{CRS\text{-}C}^{i^*}((E(L_1), \ldots, E(L_m)))$ outputs $(\lambda_1, \ldots, \lambda_m)$ with $\lambda_1 = \ldots = \lambda_m = \lambda$. Note that j^* is fix for all $i \in \mathbb{N}_m$.*

Definition 10. ($\mathcal{F}_{(\tau,\iota)\text{-Dec}}$: (τ, ι) Threshold Decryption (Dec)). *Let $E(m) \in \mathbb{C}$ be the encryption of $m \in \mathbb{P}$ under a semantically secure (τ, ι) threshold cryptosystem and let each of the participating ι parties hold a secret share used for decryption. Then, functionality $\mathcal{F}_{(\tau,\iota)\text{-}Dec}$ is given by $(m) \leftarrow \mathcal{F}_{(\tau,\iota)\text{-}Dec}(E(m))$, where at least τ out of ι parties contribute to the decryption of $E(m)$ using their secret shares.*

Two-party protocols implementing functionalities $\mathcal{F}_{\text{SC-SO}}^{\text{LT}}$ and $\mathcal{F}_{RSI}^{\omega}$ are introduced in [May12] and [FMW+14], respectively. A protocol implementing functionality $\mathcal{F}_{\text{CRS-C}}^{i^*}$ has been presented in [WMFW]. All of these protocols have been proven secure in the semi-honest model. A (τ, ι) Paillier threshold decryption protocol, $\pi_{(\tau,\iota)\text{-Dec}}$, implementing functionality $\mathcal{F}_{(\tau,\iota)\text{-Dec}}$ has been presented in [FPS01] and provides security in the semi-honest model according to [WMFW]. In this paper, we exclusively use $\pi_{\text{CRS-C}}^{i^*}$ for $i^* = 1$ and π_{RSI}^{ω} for $\omega = 0$. Consequently, we will omit these indices in the remainder of this paper.

5.2 Obliviously Evaluating Trade Party Constellation Formulas

In the following, we introduce a functionality, referred to as $\mathcal{F}_{\text{OE-TPCF}}$, for obliviously evaluating trade constellation formulas as defined in Section 4.1 and present a corresponding protocol, $\pi_{\text{OE-TPCF}}$, to securely implement functionality $\mathcal{F}_{\text{OE-TPCF}}$.

Definition 11. ($\mathcal{F}_{\text{OE-TPCF}}$: Oblivious (O) Evaluation (E) of a Trade Partner Constellation Formula (TPCF)). *Let P_i hold private input $\mathbf{q}^{(i)}$ ($i \in \mathscr{P}$) as well as a public trade partner constellation formula $\varphi \in \Phi$. Then, functionality $\mathcal{F}_{OE\text{-}TPCF}$ is given by $(E(e)) \leftarrow \mathcal{F}_{OE\text{-}TPCF}(\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\iota)}, \varphi)$ where $E(e)$ is an (ι, ι) threshold Paillier ciphertext of $e = 1$ if $[\![\varphi]\!] = 1$ and $e = 0$ otherwise.*

When designing a protocol implementing $\mathcal{F}_{\text{OE-TPCF}}$ securely, it does not suffice to evaluate each predicate $\mathcal{C}$ and $\mathcal{R}$ in φ independently and to combine the (plaintext) results via AND-operations since the involved parties would then learn whether or not a specific predicate is satisfied. Similar to [FMW+14], our approach is to intertwine the predicate evaluations instead of evaluating them independently. This allows us to evaluate φ in a privacy-preserving fashion without leaking any intermediate results.

In the following, we give a detailed description of protocol $\pi_{\text{OE-TPCF}}$ (see Protocol 1) and provide a security proof in the semi-honest model. Keeping the bartering process in mind in which $\pi_{\text{OE-TPCF}}$ is used later on, we allow that $\pi_{\text{OE-TPCF}}$ can be called by ι parties P_i ($i \in \mathscr{P}$) but only a subset of ι' parties

$P_{i'}$ with $i' \in \mathscr{P}' := \{i | i \in \mathscr{P}, \varphi \overset{\mathfrak{Q}}{\sim} TPC, TPT^{(i)} \neq (0,0)\}$ participate in the evaluation of the respective trade partner constellation formula

$$\begin{aligned}\varphi = \mathcal{C}(\mathbf{q}^{(d_1)}, \mathbf{q}^{(o_1)}) \wedge \mathcal{R}(\mathbf{q}^{(d_1)}, \mathbf{q}^{(o_1)}) \wedge \ldots \\ \wedge \mathcal{C}(\mathbf{q}^{(d_{\iota'})}, \mathbf{q}^{(o_{\iota'})}) \wedge \mathcal{R}(\mathbf{q}^{(d_{\iota'})}, \mathbf{q}^{(o_{\iota'})})\end{aligned} \tag{2}$$

with $d_j, o_j \in \mathscr{P}'$ ($j \in \mathbb{N}_{\iota'}$). Protocol $\pi_{\text{OE-TPCF}}$ iterates over each conjunction $\mathcal{C}(\mathbf{q}^{(d_j)}, \mathbf{q}^{(o_j)}) \wedge \mathcal{R}(\mathbf{q}^{(d_j)}, \mathbf{q}^{(o_j)})$ in φ ($j \in \mathbb{N}_{\iota'}$). Each such conjunction is processed sequentially and involves two parties: one that plays the role of the demander (P_{d_j}) and one that plays the role of the offerer (P_{o_j}).

W.l.o.g. we represent a commodity $c_o^{(i)}$ offered by P_i as a $|\mathscr{C}|$-bit vector $c_o^{(i)} \in \{0,1\}^{|\mathscr{C}|}$ with Hamming weight 1. More precisely, $c_o^{(i)} := (c_{o,1}^{(i)}, \ldots, c_{o,|\mathscr{C}|}^{(i)})$ where $c_{o,l}^{(i)} = 1$ if $c_o^{(i)} = c_l \in \mathscr{C}$ and $c_{o,l}^{(i)} = 0$ otherwise ($l \in \mathbb{N}_{|\mathscr{C}|}$). We use the same notation for desired commodities. The demander P_{d_1} first encrypts $c_d^{(d_1)}$ bitwise whereupon it sends the result to the offerer P_{o_1} (cf. Step 1.1.1, Protocol 1). The offerer determines the encryption of the scalar product $c_d^{(d_1)} \times c_o^{(o_1)}$ by computing ciphertext $G := \prod_{l=1}^{|\mathscr{C}|} G_l$ where G_l equals $e_l^{(d_1)} := E(c_{d,l}^{(d_1)})$ if $c_{o,l}^{(o_1)} = 1$ and otherwise 1 (cf. Step 1.3, Protocol 1). Note that there exists only one $l \in \mathbb{N}_{|\mathscr{C}|}$ such that $c_{o,l}^{(o_1)} = 1$. Next, the parties engage in $\pi_{\text{SC-SO}}^{\text{LT}}$ providing private input $\underline{q}_d^{(d_1)}$ and $\overline{q}_o^{(o_1)}$, respectively, in order to obtain an XOR-shared output indicating whether or not $\underline{q}_d^{(d_1)} < \overline{q}_o^{(o_1)}$ holds. The offerer then computes intermediate results K_1 and K_2 where, depending on w_2, one of them is set to a re-randomization of G and the other one is set to a fresh encryption of 0. K_1 and K_2 are send to the demander (cf. Step 1.5, Protocol 1). Depending on w_1, the demander sets $E(b_1)$ to a re-randomization of either K_1 or K_2, i.e., to an encryption of $[\![\mathcal{C}(\mathbf{q}^{(d_1)}\mathbf{q}^{(o_1)}) \wedge \mathcal{R}(\mathbf{q}^{(d_1)}\mathbf{q}^{(o_1)})]\!]$. He then sends $E(b_1)$ to P_{d_2} which initiates the beginning of the next iteration operating on conjunction $\mathcal{C}(\mathbf{q}^{(d_2)}\mathbf{q}^{(o_2)}) \wedge \mathcal{R}(\mathbf{q}^{(d_2)}\mathbf{q}^{(o_2)})$ and $E(b_1)$.

The iteration steps for $j = 2, ..., \iota' - 1$ proceed analogously to the one for $j = 1$ except that the demander computes

$$e_l^{(d_j)} := \begin{cases} E(b_{j-1}) \times_h c_{d,l}^{(d_j)} & \text{if } c_{d,l}^{(d_j)} = 1 \\ E(c_{d,l}^{(d_j)}) & \text{if } c_{d,l}^{(d_j)} = 0 \end{cases},$$

with $l \in \mathbb{N}_{|\mathscr{C}|}$ instead of just encrypting $c_d^{(d_j)}$ bitwise (cf. Step 1.2.1, Protocol 1).

The last iteration step, i.e., $j = \iota'$ proceeds analogously to the intermediate iteration steps for $j = 2, ..., \iota' - 1$ except that $P_{d_{\iota'}}$ sets $E(e) := E(b_{\iota'})$ [7] and broadcasts $E(e)$ (cf. Step 1.8, Protocol 1). All parties P_i ($i \in \mathscr{P}$) jointly re-randomize $E(e)$ before outputting it.[8]

[7] We write $E(m) := c$ to define $m := D(c)$, $m \in \mathbb{P}$.

[8] Note that the joint re-randomization of $E(e)$ does not contribute to the functionality of the protocol but is necessary for proving security because it decouples the randomness in the protocol output from the randomness in the protocol execution up to Step 2 (Protocol 1). In the security proof, the re-randomization is excluded from the simulation of a party's view but can be added by generating an appropriate amount of random values from $\mathbb{Z}_N^*$ and intermediate encryptions of $E(e)$ such that the views of the corrupted parties remain consistent.

Protocol 1: $\pi_{\text{OE-TPCF}}$ for obliviously evaluating trade partner constellation formulas.

1 Given φ for each j from 1 to ι':

1.1 If $j = 1$:

1.1.1 Party P_{d_1}:

1.1.1.1 Compute $e_l^{(d_1)} := E(c_{d,l}^{(d_1)})$ for $l \in \mathbb{N}_{|\mathscr{C}|}$

1.1.1.2 Send $e_1^{(d_1)}, ..., e_{|\mathscr{C}|}^{(d_1)}$ to party P_{o_1}

1.2 Else:

1.2.1 Party P_{d_j}:

1.2.1.1 Compute $e_l^{(d_j)} := \begin{cases} E(b_{j-1}) \times_h c_{d,l}^{(d_j)} & \text{if } c_{d,l}^{(d_j)} = 1 \\ E(c_{d,l}^{(d_j)}) & \text{if } c_{d,l}^{(d_j)} = 0 \end{cases}$ for $l \in \mathbb{N}_{|\mathscr{C}|}$

1.2.1.2 Send $e_1^{(d_j)}, ..., e_{|\mathscr{C}|}^{(d_j)}$ to party P_{o_j}

1.3 Party P_{o_j} computes $G := \prod_{l=1}^{|\mathscr{C}|} G_l$ where $G_l := \begin{cases} 1 & \text{if } c_{o,l}^{(o_j)} = 0 \\ e_l^{(d_j)} & \text{if } c_{o,l}^{(o_j)} = 1 \end{cases}$

1.4 Parties P_{d_j} and P_{o_j} jointly compute $(w_1, w_2) \leftarrow \pi_{\text{SC-SO}}^{\text{LT}}(\underline{q}_d^{(d_j)}, \overline{q}_o^{(o_j)})$

1.5 Party P_{o_j}:

1.5.1 Set $K_1 := \begin{cases} E(0) & \text{if } w_2 = 0 \\ Rnd(G) & \text{if } w_2 = 1 \end{cases}$

1.5.2 Set $K_2 := \begin{cases} Rnd(G) & \text{if } w_2 = 0 \\ E(0) & \text{if } w_2 = 1 \end{cases}$

1.5.3 Send K_1, K_2 to Party P_{d_j}

1.6 Party P_{d_j}:

1.6.1 Set $E(b_j) := \begin{cases} Rnd(K_1) & \text{if } w_1 = 0 \\ Rnd(K_2) & \text{if } w_1 = 1 \end{cases}$

1.7 If $j \neq \iota'$:

1.7.1 Send $E(b_j)$ to Party $P_{d_{j+1}}$

1.8 Else:

1.8.1 Set $E(e) := E(b_{\iota'})$

1.8.2 Broadcast $E(e)$

2 All parties P_i jointly re-rerandomize $E(e)$ ($i \in \mathscr{P}$)

3 Party P_i outputs $E(e)$ ($i \in \mathscr{P}$)

$c_d^{(d_j)} \stackrel{?}{=} c_o^{(o_j)}$	$Q_d^{(d_j)} \cap Q_o^{(o_j)} \stackrel{?}{\neq} \emptyset$	$G\ (j=1)$	$G\ (j=2,\ldots,\iota')$	w_1	w_2	K_1	K_2	$E(b_j)$
×	×	$E(0)$	$E(b_{j-1}\cdot 0)$	0	0	$E(0)$	$Rnd(G)$	$Rnd(K_1)$
				1	1	$Rnd(G)$	$E(0)$	$Rnd(K_2)$
×	√	$E(0)$	$E(b_{j-1}\cdot 0)$	0	1	$Rnd(G)$	$E(0)$	$Rnd(K_1)$
				1	0	$E(0)$	$Rnd(G)$	$Rnd(K_2)$
√	×	$E(1)$	$E(b_{j-1}\cdot 1)$	0	0	$E(0)$	$Rnd(G)$	$Rnd(K_1)$
				1	1	$Rnd(G)$	$E(0)$	$Rnd(K_2)$
√	√	$E(1)$	$E(b_{j-1}\cdot 1)$	0	1	$E(0)$	$Rnd(G)$	$Rnd(K_1)$
				1	0	$Rnd(G)$	$E(0)$	$Rnd(K_2)$

Table 2: Intermediate computation results of $\pi_{\text{OE-TPCF}}$.

Lemma 1. *Let P_i hold private input $\mathbf{q}^{(i)}$ ($i \in \mathscr{P}$) as well as public input φ. Then protocol $\pi_{OE\text{-}TPCF}$ securely computes functionality $\mathcal{F}_{OE\text{-}TPCF}$ in the semi-honest model.*

Proof. (CDDO.) We have to show that if $[\![\varphi]\!] = 1$, $\pi_{\text{OE-TPCF}}$ outputs $E(e)$ with $e = 1$ and otherwise $e = 0$. For $j = 1$, $b_1 = 1$ iff $c_d^{(d_1)} = c_o^{(o_1)}$ and $Q_d^{(d_1)} \cap Q_o^{(o_1)} \neq \emptyset$ hold, i.e., $[\![\mathcal{C}(\mathbf{q}^{(d_1)}\mathbf{q}^{(o_1)}) \wedge \mathcal{R}(\mathbf{q}^{(d_1)}\mathbf{q}^{(o_1)})]\!] = 1$, and otherwise $b_1 = 0$. For $j = 2,\ldots,\iota'$, $b_j = b_{j-1}$ iff $[\![\mathcal{C}(\mathbf{q}^{(d_j)}, \mathbf{q}^{(o_j)}) \wedge \mathcal{R}(\mathbf{q}^{(d_j)}, \mathbf{q}^{(o_j)})]\!] = 1$ and otherwise $b_j = 0$. The recursive computation of $E(e)$ implies that $e = 1$ iff $[\![\varphi]\!] = 1$ and otherwise $e = 0$. See Table 2 for a detailed summary of the intermediate protocol steps. (COD.) Since the output of $\pi_{\text{OE-TPCF}}$ is re-randomized and the underlying cryptosystem is semantically secure, the distribution of the output of $\pi_{\text{OE-TPCF}}$ and a fresh encryption of e are computationally indistinguishable.

(CVD.) In the following, we sketch a simulator $\mathcal{S}$ which outputs a transcript which is computationally indistinguishable from $\text{VIEW}_{I_C}^{\pi}(\widehat{x}) = (I_C, \varphi, \mathit{VIEW}_{i_1}^{\pi}, \ldots, \text{VIEW}_{i_\kappa}^{\pi})$ for each set of corrupted parties given by $I_C = \{i_1, ..., i_\kappa\}$. The detailed description of $\mathcal{S}$ is provided in Table 6 in Appendix B. By applying the modular composition theorem, it suffices for $\mathcal{S}$ to simulate the output of the sub-protocol call of $\pi_{\text{SC-SO}}^{\text{LT}}$. For a given φ, each party plays two roles: for one conjunction of the form $\mathcal{C}(\mathbf{q}^{(d_j)}, \mathbf{q}^{(o_j)}) \wedge \mathcal{R}(\mathbf{q}^{(d_j)}, \mathbf{q}^{(o_j)})$ in φ ($j \in \mathbb{N}_{\iota'}$), an involved party plays the role of the demander and for another conjunction it plays the role of the offerer. When participating in an execution of $\mathcal{F}_{\text{OE-TPCF}}$, a corrupted party P_c with $c \in I_C$ and $c \in \mathscr{P}'$ learns seven messages $m_{c,1}, \ldots, m_{c,7}$. Messages $m_{c,1} = w_1$, $m_{c,2} = K_1$, $m_{c,3} = K_2$, and a message $m_{c,7}$ send in Step 1.7.1 of Protocol 1 are learned in the role of the demander. Message $m_{c,5}$ send in Step 1.1.1.2 or Step 1.2.1.2 of Protocol 1 and a message $m_{c,6} = w_2$ are learned in the role of the offerer. Message $m_{c,4} = E(e)$ is learned independently of P_c's role. Note that P_{d_1} is an exception in that it does not learn a message $m_{d_1,7}$ since P_{d_1} initiates the evaluation of φ. A simulator $\mathcal{S}$ which simulates $\text{VIEW}_{I_C}^{\pi}(\widehat{X})$ necessarily has to check whether or not the following conditions hold:

i) For $P_c = P_{d_j}$ (resp., $P_c = P_{o_j}$) with $c \in I_C$ and $j \in \mathbb{N}_{\iota'}$ there exists an $c' \in I_C$ such that $P_{c'} = P_{o_j}$ (resp., $P_{c'} = P_{d_j}$).
ii) For $P_c = P_{d_j}$ with $c \in I_C$, $j \in \mathbb{N}_{\iota'}$ there exists an $c' \in I_C$ such that $P_{c'} = P_{d_{j-1}}$.

For the case that condition i) holds, the views of P_c and $P_{c'}$ have to be consistent, i.e., $\langle m_{c,1}\rangle$, $\langle m_{c,2}\rangle$, and $\langle m_{c,3}\rangle$ (resp., $\langle m_{c,5}\rangle$ and $\langle m_{c,6}\rangle$) have to fit to $\langle m_{c',5}\rangle$ and $\langle m_{c',6}\rangle$ (resp., $\langle m_{c',1}\rangle$, $\langle m_{c',2}\rangle$, and $\langle m_{c',3}\rangle$). Using the shared knowledge (i.e., random tapes and messages simulated so far) of the corrupted parties, $\mathcal{S}$ can simulate the required consistency (see Table 6, Appendix B). If condition i) does not hold, $m_{c,1}$, $m_{c,2}$ and $m_{c,3}$ can be simulated by $\langle m_{c_1}\rangle \leftarrow_{\$} \{0,1\}$, and $\langle m_{c,2}\rangle, \langle m_{c,3}\rangle \leftarrow_{\$} \mathbb{C}$ (resp., $\langle m_{c',5}\rangle := (\langle e_1^{(c')}\rangle, \ldots, \langle e_{|\mathscr{C}|}^{(c')}\rangle)$ with $\langle e_1^{(c')}\rangle, \ldots, \langle e_{|\mathscr{C}|}^{(c')}\rangle \leftarrow_{\$} \mathbb{C}$ and $\langle m_{c,6}\rangle \leftarrow_{\$} \{0,1\}$). This is due to the fact that in Protocol 1, K_1 and K_2 are randomized, $e_1^{(c')}, \ldots, e_{|\mathscr{C}|}^{(c')}$ are fresh encryptions, w_2 is uniformly distributed in $\{0,1\}$, and the underlying cryptosystem is semantically secure.

For the case that condition ii) holds, $m_{c,7}$ has to be simulated by using shared knowledge (i.e., random tapes and messages simulated so far) of the corrupted parties and assigning

$$\langle m_{c,7}\rangle := \begin{cases} Rnd(\langle m_{d_{j-1},2}\rangle) & \text{if } \langle m_{d_{j-1},1}\rangle = 0 \\ Rnd(\langle m_{d_{j-1},3}\rangle) & \text{if } \langle m_{d_{j-1},1}\rangle = 1 \end{cases}$$

where random bits from $\langle \mathring{r}_{d_{j-1}}\rangle$ are used to perform the randomizations. Otherwise, it suffices to set $\langle m_{c,7}\rangle \leftarrow_{\$} \mathbb{C}$. To assure that the generated views fit to each other, for each P_c, the broadcast message $m_{c,4}$ has to be simulated by assigning $\langle m_{c,4}\rangle := \langle m_{d_{\iota'},7}\rangle$ if $d_{\iota'} \in I_C$ and otherwise $\langle m_{c,4}\rangle \leftarrow_{\$} \mathbb{C}$. Table 6 in Appendix B lists the details of $\mathcal{S}$. For any I_C, it is specified how to simulate P_c's view w.r.t. the view of the other corrupted parties. First, the simulation of a corrupted party P_{d_1}'s view in both of its roles is listed (Step 1, Table 6) followed by the simulation of the other corrupted parties' views by referring to simulation steps presented for P_{d_1} (Step 2, Table 6). For the case that $c \notin \mathscr{P}'$, P_c's view consists only of its private input and $\langle m_{c,4}\rangle$ since it does not receive any message except the broadcast of $E(e)$ (cf. Step 1.8.2, Protocol 1).

Complexity. Observe that each party $P_{i'}$ ($i' \in \mathscr{P}'$) is involved in Step 1 of Protocol 1 exactly two times: in one case it is playing the role of the demander and in the other case it is playing the role of the offerer. In the former case, each party computes $|\mathscr{C}|$ ciphertexts, one homomorphic scalar multiplication (except P_{d_1}), one randomization of a ciphertext, participates in a single execution of $\pi_{\text{SC-SO}}^{\text{LT}}$, and sends $|\mathscr{C}|+1$ ciphertext. In the latter case, each party computes one randomization, one fresh encryption of 0, participates in a single execution of $\pi_{\text{SC-SO}}^{\text{LT}}$, and sends two ciphertexts. Assuming that $|\mathscr{C}|$ is fixed and φ has the structure given by Equation 2 and since the complexity of $\pi_{\text{SC-SO}}^{\text{LT}}$ is in $\mathcal{O}(1)$, the computation complexity of $\pi_{\text{OE-TPCF}}$ for each party is in $\mathcal{O}(1)$ whereas the communication and round complexities of the overall protocol are in $\mathcal{O}(\iota')$.

6 Bartering Process

In the following, we introduce our novel multi-party protocol, $\pi_{\text{ATPC-Sel}}$, for randomly selecting an actual trade partner constellation from a given public trade partner constellation set and providing each party with its local view of this actual trade partner constellation as output. We first define the underlying functionality $\mathcal{F}_{\text{ATPC-Sel}}$ followed by a detailed protocol description and a proof of

security in the semi-honest model. Additionally, we describe how the parties locally can compute their part of the actual trade from the actual trade partner constellation, i.e., to determine the actual quantities for the commodities to be traded. In Appendix A, we provide an example of how $\pi_{\text{ATPC-Sel}}$ can be used in the process of computing an actual trade from a given trade partner constellation set.

6.1 Protocol for Selecting an Actual Trade Partner Constellation

Definition 12. ($\mathcal{F}_{\text{ATPC-Sel}}$: Actual Trade Partner Constellation (ATPC) Selection (Sel).) *Let party P_i hold private input $\mathbf{q}^{(i)}$ ($i \in \mathscr{P}$). Furthermore, let $TPCS$ be an arbitrary non-empty set of trade partner constellations which is publicly known. Then, the functionality $\mathcal{F}_{ATPC\text{-}Sel}$ is defined as*

$$\left.\begin{array}{ll} (TPT_*^{(1)}, \ldots, TPT_*^{(\iota)}) & \text{if } PTPCS \neq \emptyset \\ \bot & \text{otherwise} \end{array}\right\} \leftarrow \mathcal{F}_{ATPC\text{-}Sel}(\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\iota)}, TPCS)$$

where $(TPT_*^{(1)}, \ldots, TPT_*^{(\iota)}) := ATPC \leftarrow_{\$} PTPCS \subseteq TPCS$.

In the following, $TPTS^{(i)}$ refers to the set of trade partner tuples for P_i ($i \in \mathscr{P}$) w.r.t. $TPCS$.[9]

In an ideal world where a trusted third party exists, functionality $\mathcal{F}_{\text{ATPC-Sel}}$ could be computed as follows: Each party P_i ($i \in \mathscr{P}$) sends its private input $\mathbf{q}^{(i)}$ to the trusted third party which additionally is given the public set of trade constellation tuples $TPCS$. With the knowledge of $\mathbf{Q} = \{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\iota)}\}$, the trusted third party locally computes $PTPCS \subseteq TPCS$. For the case that $PTPCS \neq \emptyset$, the trusted third party selects an actual trade partner constellation $ATPC = (TPT_*^{(1)}, \ldots, TPT_*^{(\iota)})$ uniformly at random from $PTPCS$ and sends $TPT_*^{(i)} = (x_*^{(i)}, y_*^{(i)})$ to P_i. Otherwise, the trusted third party returns $\bot$ to all parties. Note that a $(0, 0)$ output for party P_i indicates that P_i is not involved in the actual trade partner constellation while $\bot$ indicates that there exists no potential trade constellation in the given $TPCS$ at all.

In the real world, where no trusted party exists, protocol $\pi_{\text{ATPC-Sel}}$ (see Protocol 2) is executed in order to compute functionality $\mathcal{F}_{\text{ATPC-Sel}}$. Following the intuition provided in Section 4, $\pi_{\text{ATPC-Sel}}$ can be split up into the following phases:

1. *Construction Phase*: From the public set of trade partner constellations, $TPCS$, each party individually constructs the set of formulas Φ such that at the end of this phase each party holds the same set Φ.
2. *Evaluation Phase*: Each $\varphi_j \in \Phi$ is obliviously evaluated jointly by all parties P_i ($i \in \mathscr{P}$) by calling $\pi_{\text{OE-TPCF}}(\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\iota)}, \varphi_j)$ such that at the end of this phase, each party holds a vector $E(L) = (E(e_1), \ldots, E(e_{|TPCS|}))$ where $e_j = [\![\varphi_j]\!]$ ($j \in \mathbb{N}_{|TPCS|}$).
3. *Mapping Phase*: At the begin of the protocol, each party P_i ($i \in \mathscr{P}$) is given an interval $I^{(i)}$ of positive integers with at least $|TPTS^{(i)}|$ prime numbers

[9] Note that the same trade partner tuples for different TPCs are only included once in $TPTS^{(i)}$ and thus $|TPTS^{(i)}| \leq |TPCS|$.

Protocol 2: $\pi_{\text{ATPC-Sel}}$ for obliviously selecting an actual trade partner constellation.

1 Construction Phase
  1.1 Each party P_i ($i \in \mathscr{P}$) locally constructs the same set Φ from $TPCS$.
2 Evaluation Phase
  2.1 For each $\varphi_j \in \Phi$:
    2.1.1 Each party P_i participates in $(E(e_j)) \leftarrow \pi_{\text{OE-TPCF}}(\varphi_j)$
  2.2 Each party P_i sets $E(L) := (E(e_1), \ldots, E(e_{|TPCS|}))$
3 Mapping Phase
  3.1 Each party P_i:
    3.1.1 Set $S^{(i)} := \emptyset$
    3.1.2 For each $(x^{(i)}, y^{(i)}) \in TPTS^{(i)}$:
      3.1.2.1 Draw a random prime $p^{(i)}_{(x^{(i)},y^{(i)})}$ from $\mathbf{P}_{I^{(i)}} \backslash S^{(i)}$
      3.1.2.2 Update $S^{(i)} = S^{(i)} \cup \{p^{(i)}_{(x^{(i)},y^{(i)})}\}$
  3.2 Party P_ι:
    3.2.1 Set $u_j^{(\iota)} := E(p^{(\iota)}_{TPT_j^{(\iota)}})$ $(\varphi_j \overset{\mathfrak{Q}}{\sim} TPC_j)$
    3.2.2 Send $(u_1^{(\iota)}, \ldots, u_{|TPCS|}^{(\iota)})$ to $P_{\iota-1}$
  3.3 Each party $P_{i'}$ (from $i' = \iota - 1$ to 1)
    3.3.1 Compute $u_j^{(i')} := u_j^{(i'+1)} \times_h p^{(i')}_{TPT_j^{(i')}}$
    3.3.2 Send $(u_1^{(i')}, \ldots, u_{|TPCS|}^{(i')})$ to $P_{i'-1}$
  3.4 Party P_1:
    3.4.1 Set $E(L') := (E(e'_1), \ldots, E(e'_{|TPCS|})) := (u_1^{(1)}, \ldots, u_{|TPCS|}^{(1)})$
    3.4.2 Broadcast $E(L')$
4 Selection Phase
  4.1 Each party P_i participates in $((c_1^*, c_2^*)) \leftarrow \pi_{\text{CRS-C}}(E(L), E(L'))$
  4.2 For each party P_i
    4.2.1 If $c_1^* = c_2^* = \lambda$:
      4.2.1.1 Skip Steps 5 to 7
      4.2.1.2 Each party P_i outputs $\perp$
5 Decryption Phase
  5.1 Each party P_i participates in $(e_2^*) \leftarrow \pi_{(\tau,\iota)\text{-Dec}}(c_2^*)$:
6 Reverse Mapping Phase
  6.1 Each Party P_i:
    6.1.1 For each $p^{(i)}_{TPT_j^{(i)}} \in S^{(i)}$
      6.1.1.1 If $p^{(i)}_{TPT_j^{(i)}}$ divides e_2^* then $TPT_*^{(i)} := TPT_j^{(i)}$ and go to Step 7
7 Output Phase
  7.1 Each party P_i outputs $TPT_*^{(i)}$

such that for each $i, i' \in \mathscr{P}$ $(i \neq i')$, $I^{(i)}$ and $I^{(i')}$ are pairwise disjoint. Each party P_i constructs a secret table mapping each element in $TPTS^{(i)}$ to a unique prime number randomly chosen from $I^{(i)}$. More precisely, each party P_i keeps a set $S^{(i)}$ of already assigned prime numbers from $I^{(i)}$ which is initialized with $\emptyset$. P_i then maps each trader partner tuple $(x^{(i)}, y^{(i)}) \in TPTS^{(i)}$ to a prime number $p^{(i)}_{(x^{(i)},y^{(i)})} \leftarrow_{\$} \mathbf{P}_{I^{(i)}} \backslash S^{(i)}$. Subsequently, $p_{(x^{(i)},y^{(i)})}$ is added to $S^{(i)}$. Once all parties have established their mapping tables, all parties engage in the consecutive computation of an encrypted prime number product for each $\varphi_j \in \Phi$. Each party P_i contributes a single prime number $p^{(i)}_{TPT^{(i)}_j}$ to the encrypted prime number product associated with $\varphi_j \in \Phi$: First, P_ι computes $u^{(\iota)}_j = E(p^{(\iota)}_{TPT^{(\iota)}_j})$ $(j \in \mathbb{N}_{|TPCS|})$ and sends the result to $P_{\iota-1}$. Each party $P_{i'}$ from $i' = \iota - 1$ to 1 then computes $u^{(i')}_j = u^{(i'+1)}_j \times_h p_{TPT^{(i')}_j}$ and sends the results to $P_{i'-1}$, except P_1 which sets $E(L') := (E(e'_1), \ldots, E(e'_{|TPCS|})) := (u^{(1)}_1, \ldots, u^{(1)}_{|TPCS|})$ and broadcasts $E(L')$. This mapping of trade partner constellations to prime number products is the central idea of this protocol and ensures the correctness and the security of the protocol.

4. *Selection Phase*: From the previous phases, each φ_j is associated with two values $E(e_j)$ and $E(e'_j)$ where $e_j \in \{0, 1\}$ indicates whether or not φ_j is satisfied while e'_j is a product of individual prime numbers encoding the trade partner tuples of each party w.r.t. φ_j. In this phase, the parties now jointly compute $\pi_{\text{CRS-C}}$ on the common input $(E(L), E(L'))$ in order to select an entry of $E(L')$ associated with a randomly selected $\varphi_j \in \Phi_{sat}$ for the case that $\Phi_{sat} \neq \emptyset$ (i.e., $PTPCS \neq \emptyset$). Otherwise, in the case that $\Phi_{sat} = \emptyset$ (i.e., $PTPCS = \emptyset$), the parties learn of this fact. In the former case, $\pi_{\text{CRS-C}}$ returns a randomly selected pair $(c^*_1, c^*_2) \in (E(L), E(L'))$ with $c^*_1 = E(e^*_1)$ and $c^*_2 = E(e^*_2)$. In the latter case where $e_1 = \ldots = e_{|TPCS|} = 0$, $\pi_{\text{CRS-C}}(L, L')$ returns (c^*_1, c^*_2) with $c^*_1 = c^*_2 = \lambda$ which prompts each party P_i to output $\perp$ and to terminate the protocol. The purpose for this approach is to hide the number of satisfied formulas (for the case that $\Phi_{sat} \neq \emptyset$) as this could otherwise not be simulated given the inputs and outputs of the set of corrupted parties.
5. *Decryption Phase*: Each party learns e^*_2 from jointly decrypting c^*_2 together with all other parties.
6. *Reverse Mapping Phase*: Each party P_i checks which prime in $S^{(i)}$ divides e^*_2. The unique result $TPT^{(i)}_*$ determines P_i's trading partners w.r.t. $\varphi \overset{\mathbf{Q}}{\sim} ATPC$.
7. *Output Phase*: Each party P_i outputs $TPT^{(i)}_*$.

Theorem 1. *Let P_i hold $\mathbf{q}^{(i)}$ $(i \in \mathscr{P})$ and let $TPCS$ be public. Then protocol $\pi_{ATPC\text{-}Sel}$ securely computes functionality $\mathcal{F}_{ATPC\text{-}Sel}$ in the semi-honest model.*

Proof. (COD.) In order to prove COD, we distinguish two cases: (i) $TPCS \supseteq PTPCS = \emptyset$ and (ii) $TPCS \supseteq PTPCS \neq \emptyset$. For case (i), the output of $\pi_{\text{ATPC-Sel}}$ is fixed; each party outputs $\perp$. For case (ii), we have to show $ATPC = (TPT^{(1)}_*, \ldots, TPT^{(\iota)}_*)$ is selected uniformly at random from $PTPCS$.

(i) For the case that $TPCS \supseteq PTPCS = \emptyset$, the Evaluation Phase of $\pi_{\text{ATPC-Sel}}$ returns a vector $E(L) = (E(e_1), \ldots, E(e_{|TPCS|}))$ where $e_1 = \ldots = e_{|TPCS|} = 0$ since there exists no $\varphi \in \Phi$ such that $[\![\varphi]\!] = 1$. This implies that in the

Selection Phase of $\pi_{\text{ATPC-Sel}}$, $\pi_{\text{CRS-C}}(E(L), E(L'))$ returns (λ, λ). Then, each party P_i ($i \in \mathscr{P}$) outputs $\perp$ and the protocol terminates.

(ii) For the case that $TPCS \supseteq PTPCS \neq \emptyset$, the Evaluation Phase of $\pi_{\text{ATPC-Sel}}$ computes a vector $E(L) = (E(e_1), \ldots, E(e_{|TPCS|}))$ with $e_j = [\![\varphi_j]\!]$ and L has Hamming weight $|\Phi_{sat}|$. $\pi_{\text{CRS-C}}(E(L), E(L'))$, called in the Selection Phase of $\pi_{\text{ATPC-Sel}}$, returns $(E(e_1^*), E(e_2^*))$ for a random $j \in \mathbb{N}_{|TPCS|}$ such that $e_1^* = e_j' = [\![\varphi_j]\!] = 1$ ($\varphi_j \in \Phi_{sat}$) and $e_2^* = p^{(1)}_{TPT_j^{(1)}} \cdot \ldots \cdot p^{(\iota)}_{TPT_j^{(\iota)}}$. After jointly decrypting $E(e_2^*)$ in the Decryption Phase of $\pi_{\text{ATPC-Sel}}$, each party P_i obtains e_2^* and sets its $TPT_*^{(i)} := TPT_j^{(i)}$ for $p^{(i)}_{TPT_j^{(i)}} \in S^{(i)}$ where $p^{(i)}_{TPT_j^{(i)}}$ divides e_2^*. Overall, it follows that $ATPC \leftarrow_{\$} PTPCS \subseteq TPCS$.

Altogether, $\pi_{\text{ATPC-Sel}}$ provides a correct distribution of the output for both cases.

(CVD.) By separating the different phases of Protocol 2, we sketch a simulator $\mathcal{S}$ which outputs a transcript computationally indistinguishable from $\text{VIEW}^{\pi}_{I_C}(\widehat{X})$. Table 5 in Appendix B provides a detailed description of $\mathcal{S}$. The number of messages m a party P_i ($i \in \mathscr{P}$) receives when participating in $\pi_{\text{ATPC-Sel}}$ depends on the party's position in the protocol execution and on whether or not $PTPCS = \emptyset$ (cf. Table 5, Appendix B). By applying the modular composition theorem, it suffices for $\mathcal{S}$ to simulate the output of $\pi_{\text{OE-TPCF}}$, $\pi_{\text{CRS-C}}$, and $\pi_{(\tau,\iota)\text{-Dec}}$ by means of a trusted third party performing the computation of $\mathcal{F}_{\text{OE-TPCF}}$, $\mathcal{F}_{\text{CRS-C}}$, and $\mathcal{F}_{(\tau,\iota)\text{-Dec}}$, respectively. In the Evaluation Phase of $\pi_{\text{ATPC-Sel}}$, the joint outputs of the $|TPCS|$ sub-protocol calls of $\pi_{\text{OE-TPCF}}$ are simulated by setting $\langle E(L)\rangle := (\langle E(e_1)\rangle, \ldots, \langle E(e_{|TPCS|})\rangle)$ where $\langle E(e_j)\rangle \leftarrow_{\$} \mathbb{C}$ ($j \in \mathbb{N}_{|TPCS|}$). For each P_c ($c \in I_C = \{i_1, \ldots, i_\kappa\}$), $\mathcal{S}$ sets $\langle m_{c,1}\rangle := \langle E(L)\rangle$. The Mapping Phase can be simulated by performing Steps 3.1 - 3.4 of Protocol 2 for each party P_i ($i \in \mathscr{P}$). From the simulated mapping tables, $\mathcal{S}$ can simulate $(u_1^{(i)}, \ldots, u_{|TPCS|}^{(i)})$ (cf. Steps 3.2 and 3.3, Protocol 2) and $E(L')$ (cf. Step 3.4, Protocol 2). $\langle m_{c,2}\rangle$ is set to $(\langle u_1^{(c+1)}\rangle, \ldots, \langle u_{|TPCS|}^{(c+1)}\rangle)$ for $c \in I_C \backslash \{\iota\}$ while $\langle m_{c,3}\rangle$ is set to $\langle E(L')\rangle$ for $c \in I_C \backslash \{1\}$. Furthermore, $\mathcal{S}$ computes $\langle e_j'\rangle$ ($j \in \mathbb{N}_{|TPCS|}$) from the simulated mapping tables (where one of these values is used to simulate the Decryption Phase). The simulation of the Selection Phase depends on whether or not $\mathcal{F}(\widehat{X}) = \perp$. For the case that $\mathcal{F}(\widehat{X}) \neq \perp$, the output of $\pi_{\text{CRS-C}}$ is simulated by setting $\langle c_1^*\rangle, \langle c_2^*\rangle \leftarrow_{\$} \mathbb{C}$. Otherwise, $\langle c_1^*\rangle = \langle c_2^*\rangle := \lambda$. For each P_c, $\mathcal{S}$ sets $\langle m_{c,1}\rangle := (\langle c_1^*\rangle, \langle c_2^*\rangle)$. The Decryption Phase is only executed for the case that $\mathcal{F}(\widehat{X}) \neq \perp$. The output of $\pi_{(\tau,\iota)\text{-Dec}}$ is simulated by setting $\langle e_2^*\rangle := \langle e_j'\rangle$ where $j \in \mathbb{N}_{|TPCS|}$ is chosen such that $\varphi_j \overset{\mathbf{Q}}{\sim} TPC_j$ with $TPT_j^{(i_1)} = TPT_*^{(i_1)}, \ldots, TPT_j^{(i_\kappa)} = TPT_*^{(i_\kappa)}$. Note that otherwise, $\langle e_2^*\rangle$ is not consistent with $\mathcal{F}(\widehat{X})$. For each P_c, $\langle m_{c,5}\rangle$ is set to $\langle e_2^*\rangle$.

Due to the fact that the underlying cryptosystem is semantically secure, it follows that the simulated view is computationally indistinguishable from $\text{VIEW}^{\pi}_{I_C}$. Furthermore, $\mathcal{S}$ is designed in such a way that the simulated view fits to the parties output.

Complexity. Let $O_{\text{OE-TPCF}}$, $O_{\text{CRC-C}}$, and O_{Dec} denote the computation, communication, and round complexities of $\pi_{\text{OE-TPCF}}$, $\pi_{\text{CRS-C}}$, $\pi_{(\tau,\iota)\text{-Dec}}$, respectively, depending on the context. The computation complexity of $\pi_{\text{ATPC-Sel}}$ is dominated by the sub-protocol calls and $|TPCS|$ homomorphic scalar multiplications

and overall is in $\mathcal{O}(|TPCS| + |TPCS| \cdot O_{\text{OE-TPCF}} + O_{\text{CRC-C}} + O_{\text{Dec}})$. In order to determine the communication and round complexities of $\pi_{\text{OE-TPCF}}$, the staggered message exchange for computing $E(L')$ has to be considered. The communication complexity of $\pi_{\text{ATPC-Sel}}$ is in $\mathcal{O}(\iota \cdot |TPCS| + |TPCS| \cdot O_{\text{OE-TPCF}} + O_{\text{CRC-C}} + O_{\text{Dec}})$ while the round complexity is in $\mathcal{O}(\iota + |TPCS| \cdot O_{\text{OE-TPCF}} + O_{\text{CRC-C}} + O_{\text{Dec}})$.

6.2 Negotiation of Actual Quantities

In order to complete the (privacy-preserving) bartering process, i.e., for each party to compute its local view of the AT based on the $ATPC$, each party has to negotiate the actual quantities of the commodities to be traded with its trading partner. This can be done by engaging in the two-party protocol π_{RSI} for $\omega = 0$ (as introduced in [FMW+14]) with each one of its trading partners. That is for each $TPT_*^{(i)} = (x_*^{(i)}, y_*^{(i)}) \neq (0,0)$, P_i and $P_{y^{(i)}}$ participate in an execution of $(q_{c_o^{(i)}}^{(i,y^{(i)})}) \leftarrow \pi_{RSI}(Q_d^{(y^{(i)})}, Q_o^{(i)})$ where $q_{c_o^{(i)}}^{(i,y^{(i)})}$ indicates the quantity of $c_o^{(i)}$ P_i has to send to $P_{y^{(i)}}$. Note that $q_{c_o^{(i)}}^{(i,y^{(i)})}$ is chosen uniformly at random from $Q_d^{(y^{(i)})} \cap Q_o^{(i)}$ which honors the specified quantity ranges of the parties without preferring any one of them.

6.3 Optimization of ATPC Selection

Until now, we assumed that the actual trade partner constellation $ATPC$ was drawn uniformly at random from the set of potential trade partner constellations $PTPCS$. We now sketch a simple modification of protocol $\pi_{\text{ATPC-Sel}}$ which allows the private selection of an $ATPC$ with maximum *welfare* as optimization criteria, where the welfare $\mathcal{W}(\cdot)$ of a TPC is defined as the number of parties actively involved in the trade: $\mathcal{W}(TPC) := |\{TPT^{(i)} : i \in \mathscr{P}, TPT^{(i)} \in TPC, TPT^{(i)} \neq (0,0)\}|$.

The first step of our protocol modification is to introduce a prioritization of the TPCs given by $TPCS$: At the end of the Evaluation Phase (Step 2.2 in Protocol 2), the parties locally multiply the evaluation result $E(e_i)$ with $\mathcal{W}(TPC_i)$ $(\forall i \in \mathbb{N}_{|TPCS|})$ resulting in a vector $E(L) = (E(e_1) \times_h \mathcal{W}(TPC_1), \ldots, E(e_{|TPCS|}) \times_h \mathcal{W}(TPC_{|TPCS|}))$. The second step of our modification is to replace the protocol call of $\pi_{\text{CRS-C}}$ (Step 4.1, Protocol 2) by a variant of conditional random selection (also introduced in [WMFW]) which supports an integer indicator vector instead of just a binary indicator vector (cf. Definition 9). In the context of Protocol 2, this variant of $\pi_{\text{CRS-C}}$ returns $(c_1^*, c_2^*) := (E(e_{j^*}), E(e'_{j^*}))$ where $j^* \leftarrow_{\$} \{j \in \mathbb{N}_{|TPCS|} : e_j = max(e_1, \ldots, e_{|TPCS|})\}$.

Similar optimization criteria (e.g., for each TPC given by $TPCS$ a party individually determines the corresponding utility value and the welfare of a given TPC corresponds to the sum of utility values over all parties) can be integrated into protocol $\pi_{\text{ATPC-Sel}}$ analogously.

7 Conclusion & Future Work

In this paper, we presented a privacy-preserving multi-party process for determining an actual trade from a given set of trade partner constellations which is

secure in the semi-honest model. Going forward, we plan to devise and implement a privacy-preserving bartering system which uses our novel protocol at its core. Devising such a system raises further research questions like how to design the dynamics of the system, i.e., at what point the quotes of which parties should be considered for determining an actual trade. Furthermore, we plan to investigate the representation and processing of more complex commodities comprising several attributes.

References

[ABO04] E. Aïmeur, G. Brassard., and F. S. Mani Onana. Blind Sales in Electronic Commerce. In *Proceedings of the 6th International Conference on Electronic Commerce*, pages 148–157. ACM, 2004.

[AL11] G. Asharov and Y. Lindell. A Full Proof of the BGW Protocol for Perfectly-Secure Multiparty Computation. Cryptology ePrint Archive, Report 2011/136, 2011. `http://eprint.iacr.org/2011/136`.

[Ald15] L. Alderman. Trading Meat for Tires as Bartering Economy Grows in Greece, 2015. `http://www.theorderoftime.com/politics/cemetery/stout/h/pbb-24.htm`.

[AMP04] G. Aggarwal, N. Mishra, and B. Pinkas. Secure Computation of the kth-Ranked Element. In *Advances in Cryptology - EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques*, pages 40–55. Springer Berlin Heidelberg, 2004.

[Bar] BarterQuest. `http://www.barterquest.com`.

[BCD+09] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft. Secure multiparty computation goes live. In *Financial Cryptography and Data Security: 13th International Conference*, pages 325–343. Springer Berlin Heidelberg, 2009.

[Boc11] Paul Bocheck. Method, System, and Medium for Conducting Barter Transactions, 2011.

[BOGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM, 1988.

[Bra02] F. Brandt. A Verifiable, Bidder-Resolved Auction Protocol. In *5th International Workshop on Deception, Fraud and Trust in Agent Societies*, pages 18–25, 2002.

[Bri] Encyclopedia Britannica. `http://www.britannica.com/topic/barter-trade`.

[Can00] R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[CDN01] 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.

[CT12] E. De Cristofaro and G. Tsudik. Experimenting with Fast Private Set Intersection. In *Trust and Trustworthy Computing: 5th International Conference*, pages 55–73. Springer Berlin Heidelberg, 2012.

[DJ01] I. Damgård and M. Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, pages 119–136. Springer-Verlag, 2001.

[Dwo06] C. Dwork. Differential Privacy. In *Automata, Languages and Programming: 33rd International Colloquium*, pages 1–12. Springer Berlin Heidelberg, 2006.

[EA05] F. S. Mani Onana E. Aïmeur, G. Brassard. Blind Negotiation in Electronic Commerce. In *Montreal Conference on eTechnologies*, pages 1–9, 2005.

[FMW+14] 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.

[FO08] K. Frikken and L. Opyrchal. PBS: Private Bartering Systems. In *Financial Cryptography and Data Security: 12th International Conference*, pages 113–127. Springer Berlin Heidelberg, 2008.

[FPS01] P.-A. Fouque, G. Poupard, and J. Stern. Sharing Decryption in the Context of Voting or Lotteries. In *Financial Cryptography: 4th International Conference*, pages 90–104. Springer Berlin Heidelberg, 2001.

[FT98] M. Franklin and G. Tsudik. Secure Group Barter: Multi-Party Fair Exchange with Semi-Trusted Neutral Parties. In *Financial Cryptography: Second International Conference*, pages 90–102. Springer Berlin Heidelberg, 1998.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to Play ANY Mental Game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM, 1987.

[Gol09] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2009.

[HL10] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols*. Springer-Verlag Berlin Heidelberg, 2010.

[IRT] IRTA. `http://www.irta.com/`.

[KL07] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2007.

[KMRR15] S. Kannan, J. Morgenstern, R. Rogers, and A. Roth. Private Pareto Optimal Exchange. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 261–278. ACM, 2015.

[Kol06] V. Kolesnikov. *Secure Two-Party Computation and Communication*. PhD thesis, University of Toronto, 2006.

[LNRR03] N. López, M. Núñez, I. Rodríguez, 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.

[May12] D. Mayer. *Design and Implementation of Efficient Privacy-Preserving and Unbiased Reconciliation Protocols*. PhD thesis, Stevens Institute of Technology, 2012.

[min] mint.com. Barter System History: The Past and Present. `https://www.mint.com/barter-system-history-the-past-and-present`.

[NS11] T. Nishide and K. Sakurai. Distributed Paillier Cryptosystem without Trusted Dealer. In *Information Security Applications: 11th International Workshop*, pages 44–60. Springer Berlin Heidelberg, 2011.

[NSY04] J. Nzouonta, M.-C. Silaghi, and M. Yokoo. Secure Computation for Combinatorial Auctions and Market Exchanges. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1398–1399. IEEE Computer Society, 2004.

[oTE] National Association of Trade Exchanges. `http://www.natebarter.com`.

[Pai99] 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.

[Qui] D. Quilty. 36 Bartering & Swapping Websites - Best Places to Trade Stuff Online. `http://www.moneycrashers.com/best-bartering-swapping-websites/`.

[Sto] J. H. Stout. The History of Bartering and Money. `http://www.theorderoftime.com/politics/cemetery/stout/h/pbb-24.htm`.

[Tra] TradeYa. `http://www.tradeya.com/`.

[UE] U-Exchange. http://www.u-exchange.com.

[WMFW] S. Wüller, U. Meyer, F. Förg, and S. Wetzel. Privacy-Preserving Conditional Random Selection (Extended Version). `https://itsec.rwth-aachen.de/people/PPCRS_Extended.pdf/view`.

[WMFW15] S. Wüller, U. Meyer, F. Förg, and S. Wetzel. Privacy-Preserving Conditional Random Selection. In *13th Annual Conference on Privacy, Security and Trust (PST)*, pages 44–53, 2015.

[Yao82] A. C. Yao. Protocols for Secure Computations. In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, 1982.

TPC_j	TC_j	φ_j
$TPC_1 = ((5,2),(1,3),$ $(2,4),(3,5),(4,1))$	$TC_1 = (\{1,2,3,4,5\},\{(1,2),$ $(2,3),(3,4),(4,5),(5,1)\})$	$\varphi_1 = \mathcal{C}(\mathbf{q}^{(1)},\mathbf{q}^{(5)}) \wedge \mathcal{R}(\mathbf{q}^{(1)},\mathbf{q}^{(5)}) \wedge$ $\mathcal{C}(\mathbf{q}^{(2)},\mathbf{q}^{(1)}) \wedge \mathcal{R}(\mathbf{q}^{(2)},\mathbf{q}^{(1)}) \wedge$ $\mathcal{C}(\mathbf{q}^{(3)},\mathbf{q}^{(2)}) \wedge \mathcal{R}(\mathbf{q}^{(3)},\mathbf{q}^{(2)}) \wedge$ $\mathcal{C}(\mathbf{q}^{(4)},\mathbf{q}^{(3)}) \wedge \mathcal{R}(\mathbf{q}^{(4)},\mathbf{q}^{(3)}) \wedge$ $\mathcal{C}(\mathbf{q}^{(5)},\mathbf{q}^{(4)}) \wedge \mathcal{R}(\mathbf{q}^{(5)},\mathbf{q}^{(4)})$
$TPC_2 = ((4,2),(1,3),$ $(2,5),(5,1),(3,4))$	$TC_2 = (\{1,2,3,4,5\},\{(1,2),$ $(2,3),(3,5),(5,4),(4,1)\})$	$\varphi_2 = \mathcal{C}(\mathbf{q}^{(1)},\mathbf{q}^{(4)}) \wedge \mathcal{R}(\mathbf{q}^{(1)},\mathbf{q}^{(4)}) \wedge$ $\mathcal{C}(\mathbf{q}^{(2)},\mathbf{q}^{(1)}) \wedge \mathcal{R}(\mathbf{q}^{(2)},\mathbf{q}^{(1)}) \wedge$ $\mathcal{C}(\mathbf{q}^{(3)},\mathbf{q}^{(2)}) \wedge \mathcal{R}(\mathbf{q}^{(3)},\mathbf{q}^{(2)}) \wedge$ $\mathcal{C}(\mathbf{q}^{(4)},\mathbf{q}^{(5)}) \wedge \mathcal{R}(\mathbf{q}^{(4)},\mathbf{q}^{(5)}) \wedge$ $\mathcal{C}(\mathbf{q}^{(5)},\mathbf{q}^{(3)}) \wedge \mathcal{R}(\mathbf{q}^{(5)},\mathbf{q}^{(3)})$
$TPC_3 = ((5,2),(1,4),$ $(4,5),(2,3),(3,1))$	$TC_3 = (\{1,2,3,4,5\},\{(1,2),$ $(2,4),(4,3),(3,5),(5,1)\})$	$\varphi_3 = \mathcal{C}(\mathbf{q}^{(1)},\mathbf{q}^{(5)}) \wedge \mathcal{R}(\mathbf{q}^{(1)},\mathbf{q}^{(5)}) \wedge$ $\mathcal{C}(\mathbf{q}^{(2)},\mathbf{q}^{(1)}) \wedge \mathcal{R}(\mathbf{q}^{(2)},\mathbf{q}^{(1)}) \wedge$ $\mathcal{C}(\mathbf{q}^{(3)},\mathbf{q}^{(4)}) \wedge \mathcal{R}(\mathbf{q}^{(3)},\mathbf{q}^{(4)}) \wedge$ $\mathcal{C}(\mathbf{q}^{(4)},\mathbf{q}^{(2)}) \wedge \mathcal{R}(\mathbf{q}^{(4)},\mathbf{q}^{(2)}) \wedge$ $\mathcal{C}(\mathbf{q}^{(5)},\mathbf{q}^{(3)}) \wedge \mathcal{R}(\mathbf{q}^{(5)},\mathbf{q}^{(3)})$
...	...	...

Table 3: TPC_1 to TPC_3 of $TPCS$ written as trade cycle and trade partner constellation formula.

Appendix A - Example

In our example, we visualize a trade partner constellation as a directed graph which we call *trade partner constellation graph*.

Definition 13. (Trade Partner Constellation Graph.) *Given* $TPC = (TPT^{(1)}, ..., TPT^{(\iota)})$*, a trade partner constellation graph is a graph* (V, E) *with* $V := \{1, \ldots, \iota\}$ *and* $E := \{(i, y^{(i)}) | TPT^{(i)} = (x^{(i)}, y^{(i)}) \neq (0,0)\}$*. An* m*-cycle in a trade partner constellation graph is referred to as* m-trade cycle $(m{-}TC)$ *of the corresponding* TPC*.*

We consider five parties P_i ($i \in \{1, \ldots, 5\} = \mathscr{P}' = \mathscr{P}$) with private input $\mathbf{q}^{(i)}$ which aim to find a $5{-}TC$ as an actual trade partner constellation. Overall, there exist $(5-1)! = 24$ $5{-}TC$s.[10] In the following, we show how $\pi_{\text{ATPC-Sel}}$ proceeds to determine such a $5{-}TC$. As input, the protocol takes $\mathbf{Q} = \{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(5)}\}$ and $TPCS$ which consists of all the $(5-1)!$ possible $5{-}TC$s.

For each $5{-}TC_j$ with a corresponding $TPC_j \in TPCS = \{TPC_1, \ldots, TPC_{24}\}$, each party constructs $\varphi_j \overset{\mathbf{Q}}{\sim} TPC_j$ ($j \in \mathbb{N}_{24}$) (Table 3 exemplarily lists TPC_1 to TPC_3 along with the corresponding $5{-}TC$s and $TPCF$s). After obliviously evaluating each $\varphi_j \in \Phi$ by calling $\pi_{\text{OE-TPCF}}$, each party holds $E(L) = (E(e_1), \ldots, E(e_{24}))$ with $e_j = [\![\varphi_j]\!]$ and $j \in \mathbb{N}_{24}$. Based on $TPCS$, each party P_i determines its $TPTS^{(i)}$ and maps each $(x^{(i)}, y^{(i)}) \in TPTS^{(i)}$ to a unique prime number $p^{(i)}_{(x^{(i)},y^{(i)})} \in \mathbf{P}_{I^{(i)}}$. In our example we assume that $I^{(1)} = [0, 1000]$, $I^{(2)} = [1001, 2000]$ and so on. For φ_1 to φ_3, Table 4 illustrates the mapping from trade partner tuples to prime numbers for all parties.

[10] Note that the computation of the number of n-trade cycles for n parties which is $(n-1)!$ distinguishes from computing the amount of permutations of n-tuples from a domain of cardinality n without repetition which is calculated by $n!$: all permutation of tuples which can be equalized by shifting the elements to the left or right with wrap-around are isomorphic to exactly one cycle. There are $(n-1) \cdot (n-1)!$ of those tuples which can easily be verified.

	φ_1	φ_2	φ_3	...
P_1	$(5,2):73$	$(4,2):17$	$(5,2):73$	...
P_2	$(1,3):1021$	$(1,3):1021$	$(1,4):1997$	...
P_3	$(2,4):2503$	$(2,5):2837$	$(4,5):2797$	...
P_4	$(3,5):3637$	$(5,1):3559$	$(2,3):3221$	...
P_5	$(4,1):4801$	$(3,4):4583$	$(3,1):4003$	...

Table 4: Mapping trade partner tuples to prime numbers.

As described above, the parties then jointly compute the encrypted prime number products

$$E(e'_j) = E(p^{(5)}_{TPT_j^{(5)}} \cdot p^{(4)}_{TPT_j^{(4)}} \cdot p^{(3)}_{TPT_j^{(3)}} \cdot p^{(2)}_{TPT_j^{(2)}} \cdot p^{(1)}_{TPT_j^{(1)}})$$

associated with each TPC_j such that finally each party holds $E(L') = (E(e'_1), \ldots, E(e'_{24}))$. In order to privately compute an $ATPC$, parties $P_1, \ldots, P_5$ run protocol $((c_1^*, c_2^*)) \leftarrow \pi_{\text{CRS-C}}(E(L), E(L'))$ where c_1^* is an encryption of $e_1^* = 1$ for the case that there exists at least one $j \in \mathbb{N}_{24}$ such that $[\![\varphi_j]\!] = 1$ and c_2^* is the encrypted product e_2^* of prime factors mapped to the TPTs of a randomly chosen $ATPC$ from $PTPCS$. Note that for the case that there is no j such that $[\![\varphi_j]\!] = 1$, $(c_1^*, c_2^*) = (\lambda, \lambda)$ and $\pi_{\text{ATPC-Sel}}$ returns $\perp$.

For our example, we assume that $[\![\varphi_3]\!] = 1$ and that $\pi_{\text{CRS-C}}$ returns $(E(e_1^*), E(e_2^*))$ with $e_1^* = 1$ and $e_2^* = e'_3 = E(5257384086990991)$. $E(e_2^*)$ is jointly decrypted such that after decryption each party knows the plaintext e_2^*. Each party P_i then performs local trial divisions by successively dividing e_2^* by a prime number from $S^{(i)}$ until the first prime number is found which divides e_2^*. By mapping the matching prime number back to the corresponding TPT, P_i then learns its trade partners for the privately selected $ATPC$. In our example P_1, determines that $73 \in S^{(1)}$ divides e_2^* which according to P_1's mapping table implies that P_1 receives its desired commodity (of some quantity) from P_5 and has to send its offered commodity (of some quantity) to P_2. Subsequently, P_1 engages with each of P_5 and P_2 in the two-party protocol π_{RSI} to determine the actual quantities at which the commodities $c_o^{(1)}$ and $c_d^{(1)}$ will be traded.

Appendix B - Simulators

Input of $\mathcal{S}$: I_C, $TPCS$, $\widehat{X}_{I_C} = (\mathbf{q}^{(i_1)}, \ldots, \mathbf{q}^{(i_\kappa)})$, $\mathcal{F}(\widehat{X}) = \perp$ or $\mathcal{F}(\widehat{X}) = (TPT_*^{(1)}, \ldots, TPT_*^{(\iota)})$

1 Construction Phase
Nothing to simulate
2 Evaluation Phase
2.1 Simulate $E(L) = (E(e_1), \ldots, E(e_{|TPCS|}))$ by selecting $\langle E(e_j)\rangle \leftarrow_\$ \mathbb{C}$ ($j \in \mathbb{N}_{|TPCS|}$).
2.2 For each P_c ($c \in I_C = \{i_1, \ldots, i_\kappa\}$), set $\langle m_{c,1}\rangle := \langle E(L)\rangle = (\langle E(e_1)\rangle, \ldots, \langle E(e_{|TPCS|})\rangle)$.
3 Mapping Phase
3.1 For each P_i ($i \in \mathscr{P}$):
3.1.1 Follow Steps 3.1 - 3.4 of Protocol 2.
3.2 For each P_c ($c \in I_C$):
3.2.1 Set $\langle \mathring{r}_c\rangle := \langle r\rangle$ where $\langle r\rangle$ is a concatenation of $|TPCS|$ elements drawn uniformly at random from $\mathbb{Z}_{N^2}^*$ used for the simulation of P_c's computation of $u_1^{(c)}, \ldots, u_{|TPCS|}^{(c)}$ in Step 3.1.1 of $\mathcal{S}$.
3.2.2 If $c \neq \iota$:
3.2.2.1 Set $\langle m_{c,2}\rangle := (\langle u_1^{(c+1)}\rangle, \ldots, \langle u_{|TPCS|}^{(c+1)}\rangle)$ which is simulated in Step 3.1.1 of $\mathcal{S}$.
3.2.3 If $c \neq 1$:
3.2.3.1 Set $\langle m_{c,3}\rangle := \langle E(L')\rangle$ which is simulated in Step 3.1.1 of $\mathcal{S}$.
3.3 Compute $\langle e'_j\rangle$ ($j \in \mathbb{N}_{|TPCS|}$) from the simulated mapping tables in Step 3.1.1 of $\mathcal{S}$.
4 Selection Phase
4.1 If $\mathcal{F}(\widehat{X}) \neq \perp$:
4.1.1 Select $\langle c_1^*\rangle, \langle c_2^*\rangle \leftarrow_\$ \mathbb{C}$.
4.2 Else:
4.2.1 Set $\langle c_1^*\rangle = \langle c_2^*\rangle := \lambda$.
4.3 For each P_c ($c \in I_C$), set $\langle m_{c,4}\rangle := (\langle c_1^*\rangle, \langle c_2^*\rangle)$.
5 Decryption Phase
5.1 If $\mathcal{F}(\widehat{X}) \neq \perp$:
5.1.1 Set $\langle e_2^*\rangle := \langle e'_j\rangle$ where $j \in \mathbb{N}_{|TPCS|}$ s.t. $\varphi_j \sim TPC_j$ with $TPT_j^{(i_1)} = TPT_*^{(i_1)}, \ldots, TPT_j^{(i_\kappa)} = TPT_*^{(i_\kappa)}$.
5.1.2 For each P_c ($c \in I_C$), set $\langle m_{c,5}\rangle := \langle e_2^*\rangle$.
6 Reverse Mapping Phase & Output Phase
Nothing to simulate
7 If $\mathcal{F}(\widehat{X}) \neq \perp$:
7.1 If $1 \in I_C$: Set $\langle \text{VIEW}_1^\pi(\widehat{X})\rangle := (\langle \mathbf{q}^{(1)}\rangle, \langle \mathring{r}_1\rangle, \langle m_{1,1}\rangle, \langle m_{1,2}\rangle, \langle m_{1,4}\rangle, \langle m_{1,5}\rangle)$.
7.2 If $\iota \in I_C$: Set $\langle \text{VIEW}_\iota^\pi(\widehat{X})\rangle := (\langle \mathbf{q}^{(\iota)}\rangle, \langle \mathring{r}_\iota\rangle, \langle m_{\iota,1}\rangle, \langle m_{\iota,3}\rangle, \langle m_{\iota,4}\rangle, \langle m_{\iota,5}\rangle)$.
7.3 For each P_c ($c \in I_C \backslash \{1, \iota\}$), set $\langle \text{VIEW}_{c'}^\pi(\widehat{X})\rangle := (\langle \mathbf{q}^{(c')}\rangle, \langle \mathring{r}_{c'}\rangle, \langle m_{c',1}\rangle, \ldots, \langle m_{c',5}\rangle)$.
8 Else: Set $\langle \text{VIEW}_c^\pi(\widehat{X})$ ($c \in I_C$) as in Step 7 of $\mathcal{S}$ but remove $\langle m_{c,5}\rangle$ from each view.

Output of $\mathcal{S}$: $\langle \text{VIEW}_{I_C}^\pi(\widehat{X})\rangle$

Table 5: Detailed description of simulator $\mathcal{S}$ for $\pi_{\text{ATPC-Sel}}$.

Input of $\mathcal{S}$: I_C, φ, $\widehat{X}_{I_C} = (\mathbf{q}^{(i_1)}, \ldots, \mathbf{q}^{(i_\kappa)})$, $\mathcal{F}(\widehat{X}) = E(e)$

For P_c with $c \in I_C$, $c \in \mathscr{P}'$:

1 If $P_c = P_{d_1}$ ($\Rightarrow P_c \neq P_{o_1}$):
- 1.1 Simulating the *demander* role of P_c
  - 1.1.1 Select $\langle r_1 \rangle, \ldots, \langle r_{|\mathscr{C}|} \rangle \leftarrow_{\$} \mathbb{Z}_N^*$ and set $\langle r \rangle := (\langle r_1 \rangle, \ldots, \langle r_{|\mathscr{C}|} \rangle)$ (used for computing $e_1^{(d_1)}, \ldots, e_{|\mathscr{C}|}^{(d_1)}$).
  - 1.1.2 Select $\langle r_K \rangle \leftarrow_{\$} \mathbb{Z}_N^*$ (used for the randomization of K_1 or K_2).
  - 1.1.3 Select $\langle w_1 \rangle \leftarrow_{\$} \{0,1\}$ and set $\langle m_{c,1} \rangle := \langle w_1 \rangle$.
  - 1.1.4 If $o_1 \in I_C$:
    - 1.1.4.1 Compute $\langle K_1 \rangle$, $\langle K_2 \rangle$ (Step 1.5, Protocol 1) by using $\langle m_{o_1,6} \rangle$, $\langle r_{K_1} \rangle$, $\langle r_{K_2} \rangle$ from $\langle \mathrm{VIEW}_{o_1}^{\pi}(\widehat{X}) \rangle$ and the knowledge of $c_o^{(o_1)}$ and $e_1^{(d_1)}, \ldots, e_{|\mathscr{C}|}^{(d_1)}$.
  - 1.1.5 Else:
    - 1.1.5.1 Select $\langle K_1 \rangle, \langle K_2 \rangle \leftarrow_{\$} \mathbb{C}$.
  - 1.1.6 Set $\langle m_{c,2} \rangle := \langle K_1 \rangle$ and $\langle m_{c,3} \rangle := \langle K_2 \rangle$.
  - 1.1.7 If $d_{\iota'} \in I_C$:
    - 1.1.7.1 Set $\langle m_{c,4} \rangle := \langle m_{d_{\iota'},7} \rangle$.
  - 1.1.8 Else:
    - 1.1.8.1 Set $\langle m_{c,4} \rangle \leftarrow_{\$} \mathbb{C}$.
- 1.2 Simulating the *offerer* role of P_c (Let $j_c \in \{2, \ldots, \iota'\}$ s.t. $P_c = P_{o_{j_c}}$)
  - 1.2.1 Select $\langle r_{K_1} \rangle, \langle r_{K_2} \rangle \leftarrow_{\$} \mathbb{Z}_N^*$ (used for the computation of K_1 and K_2).
  - 1.2.2 Set $\langle \mathring{r}_c \rangle := (\langle r \rangle, \langle r_K \rangle, \langle r_{K_1} \rangle, \langle r_{K_2} \rangle)$.
  - 1.2.3 If $d_{j_c} \in I_C$:
    - 1.2.3.1 Compute $\langle e_1^{(d_{j_c})} \rangle, \ldots, \langle e_{|\mathscr{C}|}^{(d_{j_c})} \rangle$ (Protocol 1, Step 1.2) using $\langle r \rangle$ and $\langle m_{d_{j_c},7} \rangle$ from $\langle \mathrm{VIEW}_{d_{j_c}}^{\pi}(\widehat{X}) \rangle$.
  - 1.2.4 Else:
    - 1.2.4.1 Select $\langle e_1^{(d_{j_c})} \rangle, \ldots, \langle e_{|\mathscr{C}|}^{(d_{j_c})} \rangle \leftarrow_{\$} \mathbb{C}$.
  - 1.2.5 Set $\langle m_{c,5} \rangle := (\langle e_1^{(d_{j_c})} \rangle, \ldots, \langle e_{|\mathscr{C}|}^{(d_{j_c})} \rangle)$.
  - 1.2.6 If $d_{j_c} \in I_C$:
    - 1.2.6.1 Set $\langle w_2 \rangle$ s.t. $\langle w_2 \rangle \oplus \langle m_{d_{j_c},1} \rangle$ indicates whether or not $\underline{q}_d^{(d_{j_c})} < \overline{q}_o^{(d_{j_c})}$.
  - 1.2.7 Else:
    - 1.2.7.1 Select $\langle w_2 \rangle \leftarrow_{\$} \{0,1\}$.
  - 1.2.8 Set $\langle m_{c,6} \rangle := \langle w_2 \rangle$.
  - 1.2.9 Set $\langle \mathrm{VIEW}_c^{\pi}(\widehat{X}) \rangle := (\langle \mathbf{q}^{(c)} \rangle, \langle \mathring{r}_c \rangle, \langle m_{c,1} \rangle, \ldots, \langle m_{c,6} \rangle)$.

2 Else:
- 2.1 Simulating the *demander* role of P_c (Let $j \in \{2, \ldots, \iota'\}$ s.t. $P_c = P_{d_j}$)
  - 2.1.1 If $d_{j-1} \in I_C$:
    - 2.1.1.1 Compute $\langle E(b_{j-1}) \rangle$ (Protocol 1, Step 1.6) by using $\langle r_K \rangle$, $\langle m_{d_{j-1},1} \rangle$, $\langle m_{d_{j-1},2} \rangle$, and $\langle m_{d_{j-1},3} \rangle$ from $\langle \mathrm{VIEW}_{d_{j-1}}^{\pi}(\widehat{X}) \rangle$.
  - 2.1.2 Else
    - 2.1.2.1 Select $\langle E(b_{j-1}) \rangle \leftarrow_{\$} \mathbb{C}$.
  - 2.1.3 Set $\langle m_{c,7} \rangle := \langle E(b_{j-1}) \rangle$.
  - 2.1.4 Proceed as in Steps 1.1.1 - 1.1.8.
- 2.2 Simulating the *offerer* role of P_c (Let $j_c \in \{1, \ldots, \iota'\}$ s.t. $P_c = P_{o_{j_c}}$)
  - 2.2.1 Proceed as in Steps 1.2.1 - 1.2.2.
  - 2.2.2 If $P_c = P_{o_1}$:
    - 2.2.2.1 If $d_1 \in I_C$:
      - 2.2.2.1.1 Compute $\langle e_1^{(d_1)} \rangle, \ldots, \langle e_{|\mathscr{C}|}^{(d_1)} \rangle$ (Protocol 1, Step 1.1) using $\langle r \rangle$ from $\langle \mathrm{VIEW}_{d_1}^{\pi}(\widehat{X}) \rangle$.
    - 2.2.2.2 Else:
      - 2.2.2.2.1 Select $\langle e_1^{(d_1)} \rangle, \ldots, \langle e_{|\mathscr{C}|}^{(d_1)} \rangle \leftarrow_{\$} \mathbb{C}$.
  - 2.2.3 Else:
    - 2.2.3.1 Proceed as in Steps 1.2.3 - 1.2.5.
  - 2.2.4 Proceed as in Steps 1.2.6 - 1.2.8.
  - 2.2.5 Set $\langle \mathrm{VIEW}_i^{\pi}(\widehat{X}) \rangle := (\langle \mathbf{q}^{(c)} \rangle, \langle \mathring{r}_c \rangle, \langle m_{c,1} \rangle, \ldots, \langle m_{c,7} \rangle)$.

Output of $\mathcal{S}$: $\langle \mathrm{VIEW}_{I_C}^{\pi}(\widehat{X}) \rangle$

Table 6: Detailed description of simulator $\mathcal{S}$ for $\pi_{\text{OE-TPCF}}$.

Aachener Informatik-Berichte

This list contains all technical reports published during the past three years. A complete list of reports dating back to 1987 is available from:

`http://aib.informatik.rwth-aachen.de/`

To obtain copies please consult the above URL or send your request to:

Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen,
Email: `biblio@informatik.rwth-aachen.de`

2013-01 * Fachgruppe Informatik: Annual Report 2013

2013-02 Michael Reke: Modellbasierte Entwicklung automobiler Steuerungssysteme in Klein- und mittelständischen Unternehmen

2013-03 Markus Towara and Uwe Naumann: A Discrete Adjoint Model for OpenFOAM

2013-04 Max Sagebaum, Nicolas R. Gauger, Uwe Naumann, Johannes Lotz, and Klaus Leppkes: Algorithmic Differentiation of a Complex C++ Code with Underlying Libraries

2013-05 Andreas Rausch and Marc Sihling: Software & Systems Engineering Essentials 2013

2013-06 Marc Brockschmidt, Byron Cook, and Carsten Fuhs: Better termination proving through cooperation

2013-07 André Stollenwerk: Ein modellbasiertes Sicherheitskonzept für die extrakorporale Lungenunterstützung

2013-08 Sebastian Junges, Ulrich Loup, Florian Corzilius and Erika Ábrahám: On Gröbner Bases in the Context of Satisfiability-Modulo-Theories Solving over the Real Numbers

2013-10 Joost-Pieter Katoen, Thomas Noll, Thomas Santen, Dirk Seifert, and Hao Wu: Performance Analysis of Computing Servers using Stochastic Petri Nets and Markov Automata

2013-12 Marc Brockschmidt, Fabian Emmes, Stephan Falke, Carsten Fuhs, and Jürgen Giesl: Alternating Runtime and Size Complexity Analysis of Integer Programs

2013-13 Michael Eggert, Roger Häußling, Martin Henze, Lars Hermerschmidt, René Hummen, Daniel Kerpen, Antonio Navarro Pérez, Bernhard Rumpe, Dirk Thißen, and Klaus Wehrle: SensorCloud: Towards the Interdisciplinary Development of a Trustworthy Platform for Globally Interconnected Sensors and Actuators

2013-14 Jörg Brauer: Automatic Abstraction for Bit-Vectors using Decision Procedures

2013-16 Carsten Otto: Java Program Analysis by Symbolic Execution

2013-19 Florian Schmidt, David Orlea, and Klaus Wehrle: Support for error tolerance in the Real-Time Transport Protocol

2013-20 Jacob Palczynski: Time-Continuous Behaviour Comparison Based on Abstract Models

2014-01 * Fachgruppe Informatik: Annual Report 2014

2014-02 Daniel Merschen: Integration und Analyse von Artefakten in der modellbasierten Entwicklung eingebetteter Software

2014-03 Uwe Naumann, Klaus Leppkes, and Johannes Lotz: dco/c++ User Guide

2014-04 Namit Chaturvedi: Languages of Infinite Traces and Deterministic Asynchronous Automata

2014-05 Thomas Ströder, Jürgen Giesl, Marc Brockschmidt, Florian Frohn, Carsten Fuhs, Jera Hensel, and Peter Schneider-Kamp: Automated Termination Analysis for Programs with Pointer Arithmetic

2014-06 Esther Horbert, Germán Martín García, Simone Frintrop, and Bastian Leibe: Sequence Level Salient Object Proposals for Generic Object Detection in Video

2014-07 Niloofar Safiran, Johannes Lotz, and Uwe Naumann: Algorithmic Differentiation of Numerical Methods: Second-Order Tangent and Adjoint Solvers for Systems of Parametrized Nonlinear Equations

2014-08 Christina Jansen, Florian Göbe, and Thomas Noll: Generating Inductive Predicates for Symbolic Execution of Pointer-Manipulating Programs

2014-09 Thomas Ströder and Terrance Swift (Editors): Proceedings of the International Joint Workshop on Implementation of Constraint and Logic Programming Systems and Logic-based Methods in Programming Environments 2014

2014-14 Florian Schmidt, Matteo Ceriotti, Niklas Hauser, and Klaus Wehrle: HotBox: Testing Temperature Effects in Sensor Networks

2014-15 Dominique Gückel: Synthesis of State Space Generators for Model Checking Microcontroller Code

2014-16 Hongfei Fu: Verifying Probabilistic Systems: New Algorithms and Complexity Results

2015-01 * Fachgruppe Informatik: Annual Report 2015

2015-02 Dominik Franke: Testing Life Cycle-related Properties of Mobile Applications

2015-05 Florian Frohn, Jürgen Giesl, Jera Hensel, Cornelius Aschermann, and Thomas Ströder: Inferring Lower Bounds for Runtime Complexity

2015-06 Thomas Ströder and Wolfgang Thomas (Editors): Proceedings of the Young Researchers' Conference "Frontiers of Formal Methods"

2015-07 Hilal Diab: Experimental Validation and Mathematical Analysis of Cooperative Vehicles in a Platoon

2015-08 Mathias Pelka, Jó Agila Bitsch, Horst Hellbrück, and Klaus Wehrle (Editors): Proceedings of the 1st KuVS Expert Talk on Localization

2015-09 Xin Chen: Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models

2015-11 Stefan Wüller, Marián Kühnel, and Ulrike Meyer: Information Hiding in the Public RSA Modulus

2015-12 Christoph Matheja, Christina Jansen, and Thomas Noll: Tree-like Grammars and Separation Logic

2015-13 Andreas Polzer: Ansatz zur variantenreichen und modellbasierten Entwicklung von eingebetteten Systemen unter Berücksichtigung regelungs- und softwaretechnischer Anforderungen

2015-14 Niloofar Safiran and Uwe Naumann: Symbolic vs. Algorithmic Differentiation of GSL Integration Routines

2016-01 * Fachgruppe Informatik: Annual Report 2016

2016-02 Ibtissem Ben Makhlouf: Comparative Evaluation and Improvement of Computational Approaches to Reachability Analysis of Linear Hybrid Systems

2016-03 Florian Frohn, Matthias Naaf, Jera Hensel, Marc Brockschmidt, and Jürgen Giesl: Lower Runtime Bounds for Integer Programs

2016-04 Jera Hensel, Jürgen Giesl, Florian Frohn, and Thomas Ströder: Proving Termination of Programs with Bitvector Arithmetic by Symbolic Execution

2016-05 Mathias Pelka, Grigori Goronzy, Jó Agila Bitsch, Horst Hellbrück, and Klaus Wehrle (Editors): Proceedings of the 2nd KuVS Expert Talk on Localization

2016-06 Martin Henze, René Hummen, Roman Matzutt, Klaus Wehrle: The SensorCloud Protocol: Securely Outsourcing Sensor Data to the Cloud

2016-07 Sebastian Biallas : Verification of Programmable Logic Controller Code using Model Checking and Static Analysis

2016-09 Thomas Ströder, Jürgen Giesl, Marc Brockschmidt, Florian Frohn, Carsten Fuhs, Jera Hensel, Peter Schneider-Kamp, and Cornelius Aschermann: Automatically Proving Termination and Memory Safety for Programs with Pointer Arithmetic

* These reports are only available as a printed version.

Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.